AMIGA

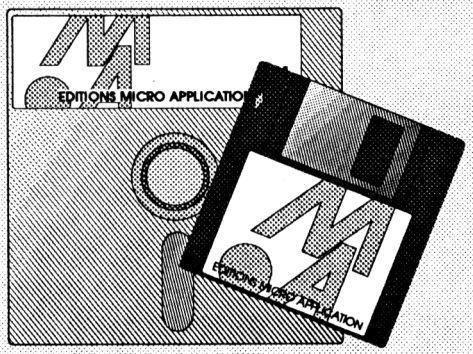
TRUCS ET ASTUCES

NOUVELLE EDITION
ENTIEREMENT
REACTUALISEE

EDITIONS MICRO APPLICATION

LIVRE DATA BECKER

Ne perdez plus de temps à saisir les programmes ...



Formats 5"1/4 ou 3"1/2 !

... procurez-vous la disquette d'accompagnement !

BON DE COMMANDE

Retournez ce bon à MICRO APPLICATION - 58 rue du fg poissonnière - 75010 Paris

Nom : ..... Prénom : .....

Adresse : .....

Code postal : ..... Ville : .....

Je désire recevoir la disquette du livre : .....

Format : ☐ 5"1/4 ☐ 3"1/2

Je joins à ce coupon :

☐ chèque de 120 FF ttc ☐ CCP de 120 FF ttc ☐ N° de carte bleue.....

Date : Signature :

Bleek
Maelger
Weltner

Trucs et astuces II Amiga

Editions Micro Application

MICRO APPLICATION
58, Rue du Faubourg Poissonnière
75010 PARIS

© Reproduction interdite sans l'autorisation de
MICRO APPLICATION

'Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (Loi du 11 Mars 1957, article 40, 1er alinéa).

Cette représentation ou reproduction illicite, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La Loi du 11 Mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration'.

ISBN : 2-86899-220-x

© 1989 Data Becker GmbH
Merowingerstrasse, 30
4000 Düsseldorf

© 1989 MICRO APPLICATION
58 Rue du Faubourg Poissonnière
75010 PARIS

Auteurs : Bleek, Maelger, Weltner
Traduction française assurée par Isabelle Murru

Collection dirigée par Mr Philippe OLIVIER
Edition réalisée par Frédérique BEAUDONNET

Amiga est un modèle déposé de Commodore-Amiga Inc.
Intuition est une marque déposée de Commodore-Amiga Inc.
Workbench est une marque déposée de Commodore-Amiga Inc.
C64 est un modèle déposé de Commodore Electronics Limited
Commodore est une marque déposée de Commodore Electronics Limited
BASIC est une marque déposée de Microsoft Corporation
Lattice C est une marque déposée de Lattice Corporation
Aztec C est une marque déposée de Manx Software Systems
HiSoft, Devpac et HiBasic sont des marques déposée de HiSoft
IBM est une marque déposée de International Business Machines
Atari ST est un modèle déposé de Atari Corporation

Préface

Chaque nouveau système est un événement à lui tout seul. Tout d'abord cela montre que, pour notre grand plaisir, l'Amiga sait parfaitement évoluer et d'autre part que Commodore tient à conserver l'image d'enfant prodige qui sied si bien à notre machine fétiche. C'est pourquoi nous avons accueilli avec satisfaction la sortie du Kickstart et du Workbench 1.3, preuves indéniables, s'il en fallait, des capacités innovatrices de la firme américaine.

Commodore a donc enrichi ces versions de nouvelles commandes, d'applications plus perfectionnées et d'un système d'exploitation encore plus performant.

Fidèles à notre tradition, nous nous sommes donc mis en quête des nouveautés et des originalités que présentent le Kickstart et le Workbench 1.3. Nous les avons dépouillés, analysés, en répertoriant parmi ces innovations tout ce dont l'utilisateur gagnerait à connaître. Aussi les différences entre les systèmes 1.2 et 1.3 sont décrites dans le détail.

Trucs et astuces est donc le fruit de nos recherches et de nos expérimentations. Le présent ouvrage a été conçu dans le même esprit que la précédente édition. Vous y trouverez des informations capitales, celles que l'on ne devine pas nécessairement par soi-même, bref ces petites touches de "différences" qui apporteront à vos applications une indéniable note d'originalité. Et bien sûr tout un éventail de programmes que vous pourrez adapter rapidement à vos propres besoins.

Ce livre explore les Workbench 1.2 et 1.3 dans les moindres détails. Du logiciel au matériel en passant par le système d'exploitation, nous n'avons toujours eu qu'une idée en tête : ne négliger aucun des aspects qui vous permettront par la suite d'utiliser votre Amiga au maximum de ses capacités.

Sommaire

1. Nouveaux trucs et astuces 9

1.1. Astuces pour le CLI 9
1.2. Astuces pour AmigaBASIC 15
1.3. Astuces d'impression 20
1.4. Astuces de travail avec Amiga 22
1.5. Les cachettes des messages secrets 23

2. Trucs et astuces de programmation en AmigaBasic ... 27

2.1. Une fenêtre Intuition 28
2.1.1. Les gadgets 32
2.1.2. Contour et texte d'un Gadget 39
2.1.3. Des Gadgets familiers 45
2.1.4. Les Gadgets de sélection 51
2.1.5. Les ascenceurs 57
2.1.6. Les Gadgets proportionnels 64
2.2. Le gestionnaire de fichiers 70
2.2.1. Utilisation du gestionnaire de fichiers 84
2.2.2. Modifier des fichiers à l'aide du moniteur 87
2.2.3. Modification de AmigaBASIC 87
2.3. Accéder à AmigaDOS à partir d'AmigaBASIC 88
2.3.1. Accès aux fichiers sous AmigaDOS 90
2.3.2. Informations sur les fichiers, les répertoires et le disque 98
2.3.3. Gestion directe des fichiers 105
2.4. Bibliothèques et applications 108
2.4.1. Le gestionnaire de répertoire 108

3. Le langage machine 115

3.1. La division par zéro 116
3.2. Alerte aux virus 120
3.2.1. L'arme suprême contre les virus 121
3.3. Le langage machine et BASIC 125
3.3.1. Assembleur et programmes C à partir de BASIC 128
3.3.2. Perfectionnement de BASIC : ColorCycle 131
3.3.3. Inactiver la souris 134

4. Incursion dans le matériel 137

4.1. Mise hors service de l'extension de la mémoire 138
4.1.1. La carte 2000A 139
4.1.2. La carte de circuit imprimé de l'Amiga 500 141
4.2. Contrôle des lecteurs de disquette par commutateur 142

4.3. Installation d'un MC 68010 143
4.4. Le ronflement du ventilateur 146
4.5. Les nouveaux processeurs 147
4.5.1. Le 68010 : la puissance à bas prix 147
4.5.2. Le 68012 : grande capacité, bas coût. 149
4.5.3. Des processeurs monstres : 68020, 68030 et 6888x 150

5. Workbench 1.3 153

5.1. Utilisation de Mount 153
5.1.1. Renommer les commandes 156
5.1.2. Rien ne se perd, tout se transforme 157
5.2. Améliorations apportées à DOS 1.3 158
5.2.1. Le périphérique PIPE 159
5.2.2. Le périphérique Speak 160
5.2.3. Le périphérique NewCon 161
5.2.4. Le FastFileSystem 161
5.2.5. FFS et les disques durs 162
5.2.6. Les nouvelles bibliothèques de fonctions mathématiques ... 163

6. Le printer.device 167

6.1. Sélection des paramètres d'impression 167
6.2. Les impressions graphiques au moyen du printer.device 174
6.2.1. Application de Hardcopy 179

7. Les disquettes Workbench et Extras 183

7.1. Preferences 183
7.1.1. Lecture et définitions des paramètres de Preferences 184
7.1.2. La nouvelle version de Preferences (Version 1.3) 190
7.1.3. Le tiroir Utilities dans 1.3 196
7.1.3.1. Les autres utilitaires de Workbench 1.3 201
7.2. Les outils de la disquette Extras 202

8. Personnalisation du Workbench 209

8.1. Les préparatifs 210
8.2. Les premières étapes de la modification d'un texte 212
8.2.1. Les messages de démarrage et la fenêtre AmigaDOS 212
8.2.2. Modification de la barre de titre et des menus. 216
8.2.3. Du nouveau dans les rubriques de menus et dans les messages .. 216
8.3. Guide des messages de Kickstart 220

Annexe A : Messages d'erreur 223

Annexe B : Les raccourcis CLI 227

Annexe C : Séquences Escape d'impression 229

Annexe D : Les codes de Guru Meditation 233

Index 239

Chapitre 1

Nouveaux trucs et astuces

Ce chapitre contient un certain nombre de petites suggestions et informations qui constituent une sorte de "mini-guide des trucs et astuces". Il comprend des informations spécifiques aux Workbench 1.2 et 1.3. Nous espérons que ces trucs et astuces rendront vos sessions avec Amiga plus productives et à la fois plus agréables.

1.1. Astuces pour le CLI

#1 : Sauvegardes automatiques

Ne vous est-il jamais arrivé de modifier le fichier Startup-Sequence à l'aide de Ed, de quitter Ed, et de vous apercevoir que sans la ligne que vous venez juste d'effacer votre Amiga ne pourra être initialisée ? Si cela se produit, pas d'affolement. Ed crée toujours une sauvegarde automatique du dernier fichier édité qu'il place dans le répertoire t:(temporaire). Le répertoire T: conserve les fichiers temporaires sur le disque pour le cas où vous auriez fait des dégâts dans le fichier en cours d'édition.

Pour récupérer l'ancien fichier Startup-Sequence, copiez le fichier ed-backup situé dans le répertoire T: sous le nom Startup-Sequence dans le répertoire S:. La ligne de commande se présente de la façon suivante :

```
Copy :T/ed-backup to S:Startup-Sequence¶
```

Remarque : Ne copiez pas le répertoire T: sur le RAM-disk car une réinitialisation effacera le répertoire et vous ne pourrez plus y accéder en cas d'incident.

```
#2 : Interruption de la sortie CLI
```

Vous pouvez interrompre le défilement du texte dans la fenêtre CLI (lorsque vous exécutez, par exemple, les commandes List, Dir ou Type, ou toute autre commande qui affiche une longue liste de données d'un seul tenant) en enfonçant une touche quelconque. CON: bloque l'écran jusqu'à ce que vous appuyiez sur la touche <Backspace>. A ce moment, le défilement du texte reprend. Un moyen simple d'interrompre et de reprendre le défilement d'un répertoire est d'appuyer alternativement sur la barre d'espacement puis sur <Backspace>.

#3 : Modifier l'aspect du CLI

Peut-être avez-vous déjà remarqué que certains disques affichent un texte de différentes couleurs et styles de caractères au cours de l'initialisation ? Il n'y a aucun tour de passe-passe là-dessous. Vous pouvez modifier les couleurs à votre gré. Il en va de même du texte qui peut apparaître en italiques, en gras ou en souligné. C'est la commande ECHO qui permet de réaliser ces modifications.

Pour obtenir ces effets, des caractères de contrôle sont nécessaires. Il informent l'ordinateur que les caractères qu'il rencontre sont des caractères d'échappement (ESC) qui doivent être exécutés et non pas imprimés. Ces caractères de commande sont inscrits entre guillements et débutent toujours par "*e". "*e" représente la touche <ESC> et indique un changement de code. Cette séquence est suivies de caractères et de nombres. Certains caractères sont séparés par des points virgules et intègrent la séquence de contrôle. Le chiffre 4, par exemple, indique le soulignement et le nombre 42, un fond noir (les différentes options possibles et les valeurs correspondantes sont indiquées dans le tableau ci-dessous). La séquence de contrôle se termine par un m et le texte que l'on désire afficher.

Faites l'essai suivant : créez le fichier UNDERLINE avec ED ou un autre éditeur. Entrez ce qui suit :

```
echo"e[4mUNDERLINE on"¶
echo"e[0mNormal"¶
```

Sauvegardez le fichier et quittez l'éditeur. Maintenant, entrez Execute Underline et appuyez sur <Return>. Et voilà le résultat :

```
Underline¶
Normal¶
```

Le tableau suivant dresse la liste des valeurs de la séquence de contrôle :

Style	Valeur	Remarque
normal	0	
gras	1	
italique	3	
souligné	4	
vidéo inverse	7	
Couleur du 1er plan	**Valeur**	**Remarque**
Normal	30	Sélectionné avec Preferences
Blanc	31	
Noir	32	
Orange	33	
Couleur de l'arrière plan	**Valeur**	**Remarque**
Normal	40	Sélectionné avec Preferences
Blanc	41	
Noir	42	
Orange	43	

#4 : Combinaisons de touches avec <Ctrl>

Les combinaisons de touches permettent de réaliser un grand nombre d'actions. La plupart du temps, c'est la touche <Ctrl> qui est utilisée en combinaison avec d'autres touches. Si l'on maintient la touche <Ctrl>

enfoncée et que l'on appuie sur une autre touche cette action a une incidence sur l'aspect de l'écran ou le déroulement d'un programme. Cette combinaison de touche est généralement appelée la touche contrôle.

Les quatre premières touches contrôle permettent d'interrompre l'exécution de nombreux programmes.

<Ctrl><C>	Interrompt une commande CLI
<Ctrl><D>	Interrompt un script en cours d'exécution.
<Ctrl><E>	Exécute une interruption de priorité supérieure.
<Ctrl><F>	Exécute une interruption de priorité supérieure.

Voici maintenant quelques unes des combinaisons les plus usitées pour contrôler l'écran.

<Ctrl><G>	Clignotement de l'écran (sans signal sonore).
<Ctrl><H>	Même fonction que la touche <Delete>.
<Ctrl><I>	Même fonction que la touche <Tab>.
<Ctrl><J>	Nouvelle ligne et mise en attente de validation.
<Ctrl><K>	Même fonction que la touche <Curseur Haut>.
<Ctrl><L>	Supprime la fenêtre (Même fonction que <Esc><c>).
<Ctrl><M>	Même fonction que la touche <Return>.
<Ctrl><N>	Active un nouveau jeu de caractères.
<Ctrl><O>	Restitue le jeu de caractères initial(Même fonction que <Return>.
<Ctrl><X>	Efface le contenu de la ligne courante.

#5 : Mieux copier !

La commande Copy permet de copier des fichiers à partir du CLI. Il existe plusieurs variantes à cette commande qui sont assez mal documentées (si toutefois il existe une quelconque documentation sur ce sujet). La commande copy vous permet, entre autres, de copier plusieurs fichiers qui n'ont aucun rapport les uns avec les autres et qui ne peuvent être regroupés grâce aux caractères de substitution (# et ?).

La solution est très simple : vous pouvez copier des fichiers qui portent le même nom mais qui ont une extension différente à l'aide du caractère barre horizontale (|). Dans l'exemple suivant, trois fichiers du même nom (test.c, test.h et test.o) sont copiés sur le RAM-Disk.

```
Copy C:test.(c|h|o) TO RAM:¶
```

De manière différente, le nom du fichier peut être considéré comme une extension ce qui permet de réaliser plusieurs copies sans répéter à chaque fois le nom de la commande. Dans l'exemple suivant, les trois commandes CLI : Dir, List et Rename sont copiées du répertoire C: sur le répertoire C du RAM-Disk.

```
Copy C:(DIR|LIST|RENAME) TO RAM:C¶
```

#6 : Fenêtres CLI et fichiers scripts

Lorsque vous ouvrez une nouvelle fenêtre CLI, vous pouvez lui attribuer les dimensions et le titre de votre choix. De plus, vous pouvez affecter la nouvelle fenêtre à l'exécution d'un fichier script de la façon suivante :

```
NewCLI Outputdevice: Script-file¶
```

Dans la version 1.2, l'argument de Outputdevice était CON:, suivi des dimensions de la fenêtre. Le fichier script est exécuté après l'ouverture de NewCLI. Un fichier-script peut être exécuté en parallèle avec un autre processus lancé dans une autre fenêtre.

Les utilisateurs en possession du Workbench 1.3 doivent utiliser NEWCON: comme argument de Outputdevice (de cette façon les fonctions d'édition de Shell sont maintenues dans la nouvelle fenêtre), ou NewShell sans spécifier de périphérique de sortie.

#7 : Les icônes de disque

L'aspect de Workbench 1.3 diffère légèrement de la version précédente. Tous les anciens périphériques sont représentés par les anciennes icônes : c'est le cas du RAM-disk et de RAMB0 L'icône qui symbolise la disquette Workbench peut aussi servir à représenter le RAM-Disk et RAMB0. Il suffit, pour cela, d'ajouter deux brèves commandes Copy dans la Startup-Sequence. Les lignes de commande sont les suivantes :

```
Copy Disk.info to RAM:¶
Copy Disk.info to RAD:¶
```

En insérant les commandes Copy avant LoadWB, vous obtiendrez une icône uniforme. Cette méthode peut être appliquée aussi aux autres disques.

#8 : Une fenêtre, deux CLI.

Deux CLI ne peuvent pas fonctionner simultanément dans deux fenêtres différentes. Il est cependant possible d'avoir deux CLI dans une seule fenêtre. La commande suivante permet de réaliser cela très simplement :

```
NewCLI *¶
```

Cette commande dirige la zone de sortie de NewCLI vers la fenêtre en cours d'utilisation sans ouvrir de seconde fenêtre. Vous pouvez ainsi passer d'une tâche à l'autre sans avoir à changer de fenêtre de travail, manipulation qui peut devenir fastidieuse à la longue. Dans ce cas, utilisez la commande Run au lieu de Execute qui convient mieux à un appel de programme direct.

#9 : Une fenêtre CLI sans cadre

Certains programmes tentent de supprimer le cadre de la fenêtre CLI à l'aide de méthodes compliquées. Les auteurs de ces programmes semblent avoir oublié que le CLI est créé à partir du périphérique console. Console contient des commandes qui permettent de modifier à loisir l'aspect du cadre de la fenêtre à l'aide de séquences d'échappement (ESC).

Séquence	Explication
<Esc> [n u	Largeur en caractères de la fenêtre CLI(n).
<Esc> [n x	Marge gauche de la fenêtre CLI en pixels (n).
<Esc> [n y	Distance en pixels du bord supérieur de la fenêtre (n).
<Esc> [n t	Nombre de lignes de la fenêtre CLI (n).
<Esc> c	Réinitialisation de la fenêtre.

La commande Echo est utilisée pour appeler chaque séquence. Il est possible de chaîner plusieurs séquences avec des points virgules. Pour afficher le cadre après un <Esc>c, il faut redimensionner la fenêtre. Voici un exemple:

```
;Cette séquence supprime le cadre de la fenêtre¶
echo "*e[80u*e[0x*e[0y*e[31t"¶
;Appuyer sur <Ctrl><L><Return> pour obtenir une
fenêtre sans cadre¶
¶
;Cette séquence remet le système à son état initial¶
echo "*ec"¶
```

#10 : Messages personnels

Généralement le programme de démarrage affiche le numéro de version du Workbench et la date en cours. Vous pouvez ajouter des messages personnels à ces informations si vous le désirez.

Le problème est que lorsque la commande Echo affiche le texte, elle doit être systèmatiquement rechargée, ce qui prend du temps. Vous pouvez accélerer la procédure en tirant parti des fonctions multitâches que vous offre l'Amiga : écrivez la totalité du texte dans un fichier qui s'affichera à l'écran au moment de la séquence de démarrage.

Entrez la ligne suivante dans le fichier contenant votre programme de démarrage afin d'appeler le fichier texte que vous voulez voir apparaitre à l'écran.

```
Run Type Textfile¶
```

1.2. Astuces pour AmigaBASIC

#11 : Fermeture à l'aide d'icône standard

AmigaBASIC utilise une icône particulière pour représenter l'ensemble des programmes créés sous AmigaBASIC, ainsi que pour les fichiers de données créés à partir des programmes AmigaBASIC. Il existe deux façons de modifier cela :

Vous pouvez éditer l'icône du programme BASIC après avoir achevé l'écriture de votre programme à l'aide de l'éditeur d'icônes. Attribuez cette nouvelle icône au programme et copiez la dans le répertoire approprié. L'astuce consiste à définir le drapeau de suppression dans le fichier Info de telle sorte que le fichier Info ne puisse être effacé. Ainsi, l'icône ne sera pas écrasée la prochaine fois que vous sauvegarderez le programme.

La méthode est plus simple lorsqu'il s'agit des fichiers de données. L'icône de ces fichiers doit être en rapport avec le programme concerné (par exemple, les fichiers texte écrits dans NotePad ont une icône Note). Tout d'abord, il faut dessiner l'icône voulue avec l'éditeur d'icône. Cette icône est enregistrée dans le même répertoire que le programme principal. Au début vous lirez l'icône à l'aide de la fonction GetDiskObject. Chaque fois que vous sauvegarderez ce fichier, l'cône sera enregistrée sous le même nom. Votre icône remplacera l'icône d'AmigaBASIC. Voici un programme qui attribue l'icône Shell à un programme BASIC. Modifiez les noms des disques (T&A2:Chapitre1) et des tiroirs pour adapter l'ensemble à votre propre configuration.

Début du programme

```
REM ICONInstall dans le tiroir Chapitre 1 de la
disquette T&A2¶
LIBRARY "T&A2:bmaps/icon.library" :REM utiliser
ConvertFD¶
DECLARE FUNCTION GetDiskObject& LIBRARY¶
DECLARE FUNCTION PutDiskObject& LIBRARY¶
Filename$ = "SYS:Shell"+CHR$(0)¶
DiskAdr& = GetDiskObject&(SADD(Filename$))¶
```

Après avoir fermé le fichier

```
File$="T&A2:Chapitre1/IconInstall"¶
File$=File$+CHR$(0)¶
status = PutDiskObject&(SADD(File$), DiskAdr&)¶
PRINT status¶
```

#12 : Des programmes modulaires

Vos programmes doivent, autant que faire se peut, avoir une structure modulaire. Vos listing n'en seront que plus faciles à lire et à modifier. La modularité de vos programmes présente, en outre, l'avantage de pouvoir les intégrer à des programmes ultérieurs. Vous gagnerez ainsi beaucoup de temps. Il existe deux manières d'incorporer des modules de programmes à d'autres listings:

La première est de charger l'interpréteur BASIC et de copier le module à partir du programme source dans le clipboard. Ce dernier pourra ensuite être inséré dans un second programme en BASIC. Pour ce faire, il faut charger le second programme et réinsérer la partie de programme enregistrée dans le listing à l'aide de la fonction Paste.

La seconde méthode implique que vous sauvegardiez le module de programme dans un fichier. Il est indispensable de réaliser la sauvegarde sous format ASCII. Vous pouvez ensuite intégrer le nouveau module dans un programme à l'aide de:

```
Merge Filename¶
```

Ce faisant vous économiserez du travail et vous pourrez constituer une bibliothèque composée de modules de fonctions indépendants.

#13 : Modification des titres de fenêtre et d'écran

Dans AmigaBASIC, vous ne pouvez pas modifier après coup le nom attribué à une fenêtre. La bibliothèque Intuition vous offre une solution :

Ouvrez la bibliothèque au début du programme. Vous pourrez préciser le nom de la fenêtre et de l'écran en appelant SetWindowTitle(). Dans le programme donné, ci-dessous, à titre d'exemple, la fenêtre BASIC est rebaptisée test et l'écran Workbench, Ecran. Soulignons ici que les variables utilisées pour fournir des paramètres à CALL doivent se terminer par des octets nuls (Chr$).

```
LIBRARY "T&A2:bmaps/intuition.library":'indiquer le
chemin¶
'd'accès à votre bibliothèque Intuition.¶
CALL SetTitle("test","Ecran")¶
END¶
¶
SUB SetTitle(WinNam$, ScrNam$) STATIC¶
 WinNam$ = WinNam$ + CHR$(0)¶
 ScrNam$ = ScrNam$ + CHR$(0)¶
 CALL SetWindowTitles(WINDOW(7), SADD(WinNam$),
SADD(ScrNam$))¶
END SUB¶
```

#14 : Plus de mémoire

En règle générale, dans AmigaBASIC, la commande Clear permet d'augmenter l'espace mémoire. Malheureusement, cette commande ne produit pas le résultat escompté si l'espace recherché n'est pas disponible (c'est un défaut d'AmigaBASIC). La commande cherche d'abord à allouer un nouvel espace mémoire puis elle tente de libérer un espace mémoire utilisé. Lorsque vous désirez modifier votre espace mémoire un tant soit peu, il faut que l'espace disponible soit deux fois plus important que l'espace demandé, ce qui n'est pas souvent le cas. Il n'existe qu'une seule méthode pour pallier à cela. Il s'agit tout simplement de définir la taille la plus petite, puis d'entrer la taille voulue.

```
Clear, 1024¶
Clear, 50000¶
```

Cette méthode est déconseillée lorsque vous êtes en mode programme. Dans ce cas, il est préférable d'utiliser ce qui suit :

```
Clear, 25000-FRE(0) 'mémoire nécessaire au programme
uniquement¶
Clear, FRE(-1)-50000 'totalité de la mémoire
disponible- sécurité¶
```

#15 : Edition des programmes en BASIC

L'éditeur BASIC n'est pas toujours à la hauteur de sa tâche. Le défilement horizontal se fait avec peine, quant au defilement vertical, ce n'est pas un modèle de rapidité. De plus, la fonction recherche est inexistante. AmigaBASIC apporte une solution à ces problèmes. Vous pouvez sauvegarder les programmes sous la forme d'un texte ASCII à l'aide de Save"nom",a puis les éditer avec un traitement de texte. De cette façon, les programmes peuvent aussi être transférés sur d'autres ordinateurs.

Les traitements de texte offrent bien plus de possibilités que l'éditeur AmigaBASIC. L'environnement multitâches d'Amiga permet d'exploiter simultanément BASIC et un traitement de texte. Vous pouvez ainsi déplacer rapidement le programme édité vers AmigaBASIC pour le tester. L'AmigaBASIC ne gère pas le clipboard de façon standard mais utilise un fichier intitulé RAM:BasicClip pour ses opérations de Couper/Coller/Copier. Vous pouvez également manipuler ce fichier avec un traitement de textes.

#16 : AmigaBASIC plus rapide

La rapidité d'AmigaBASIC peut être améliorée. Il faut savoir que lorsque vous démarrez AmigaBASIC à partir de Workbench, les priorités ne sont pas définies de manière à permettre un fonctionnement optimal d'AmigaBASIC.

Deux fonctions de Exec.library apportent une solution à ce problème. Grâce à SetTaskPri(), vous pouvez visualiser les priorités affectées à vos tâches ainsi qu'à BASIC et sélectionner une valeur comprise entre 1 et 127. Plus la valeur est élevée, plus Amiga est rapide. Cela signifie que le processeur accorde moins de temps à l'exécution des autres programmes ou tâches. Cette méthode est excellente pour les longs calculs et pour les sorties graphiques.

Tout d'abord, il faut accéder à Exec.library. Ensuite, il faut rechercher la tâche à l'aide de FindTask() et modifier les priorités avec SetTaskPri(). Une fois le programme BASIC terminé, il est indispensable de redéfinir les priorités afin que les autres programmes puissent être exécutés à leur tour.

Début du programme

```
LIBRARY "T&A2:bmap/exec.library"
DECLARE FUNCTION FindTask& LIBRARY
BASICTask& = FindTask&(0)
CALL SetTaskPri(BASICTask&, 80)
```

Fin du programme

```
CALL SetTaskPri(BASICTask&, 0)¶
```

#17 : Suppression du dépassement du tampon de ligne.

Avez-vous déjà été confronté à ceci ? Quelquefois, l'éditeur AmigaBASIC vous empêche de revenir sur une ligne à l'aide de la touche <Backspace>. Il génère le message d'erreur "line buffer overflow". Pour contourner cela, il suffit de sélectionner un espace avec la souris et de le supprimer à l'aide de <Amiga><X>. Les choses reviennent ensuite à la normale.

#18 : Réinitialisation (KickStart 1.2 seulement)

Si vous désirez que votre programme BASIC réinitialise l'Amiga, utilisez le code suivant (avec prudence):

```
bye = 16515072¶
CALL bye¶
```

1.3. Astuces d'impression

#19 : Modification de l'imprimante sans passer par Preferences

Peu de personnes savent que le driver d'imprimante est un programme autonome qui peut s'autodémarrer. Cependant sa nature spécifique ne permet pas que l'on y accède par l'intermédiaire du CLI. Workbench sait à quel endroit de la mémoire le driver doit être placé. Copiez le driver à partir du répertoire DEVS:Printers dans le répertoire principal du disque Workbench. Utilisez ensuite une icône de type Tool comme l'icône d'horloge ou du CLI. Baptisez ces icônes du nom des drivers d'imprimante. Lorsque vous cliquez sur l'icône à partir de Workbench, le driver d'imprimante correspondant s'initialise. Vous trouverez ci-dessous les commandes CLI du driver de l'imprimante Epson.

```
1> copy sys:devs/printers/epson sys:epson¶
1> copy CLI.info Epson.info¶
```

#20 : Interêt de l'impression indirecte

Certaines personnes accèdent directement aux fonctions de l'imprimante à partir d'un programme en BASIC à l'aide des périphériquess PAR: et SER:. Normalement toutes les sorties impression doivent se faire par le périphérique PRT:

Commodore-Amiga met en oeuvre des séquences génériques qui contrôlent l'impression. Le driver d'imprimante convertit les séquences propres au type d'imprimante qu'il gère. De la sorte, les routines d'impression sont valables d'une imprimante à l'autre. Le tableau suivant

indique comment obtenir différents style d'impression. Toutes les séquences débutent par <Esc>. Il est préférable de définir <Esc> à l'aide d'une chaîne de caractères comme dans l'exemple ci-dessous:

```
ESC$=CHR$(27)¶
```

CHR$ représente la touche <Esc>. Cette chaîne de caractères indique à l'imprimante que des codes spéciaux doivent être préparés pour contrôler l'imprimante ou d'autres périphériques.

Liste des styles d'impression et des séquences Escape

Style	Séquence
Italiques	activé:ESC[3m désactivé:ESC[23m
Gras	activé:ESC[1m désactivé:ESC[22m
Souligné	activé:ESC[4m désactivé:ESC[24m
Police Elite	activé:ESC[2w désactivé:ESC[1w
Mode condensé	activé:ESC[4w désactivé:ESC[3w
Mode étendu	activé:ESC[6W désactivé:ESC[5W
NLQ	activé:ESC[2"z désactivé:ESC[1"z
Proportionnel	activé:ESC[2p désactivé:ESC[1p
Exposant	activé:ESC[2v désactivé:ESC[1v
Indice	activé:ESC[4v désactivé:ESC[3v

Une séquence de commande peut être transmise à l'imprimante en même temps que le texte à l'aide de LPRINT.

```
LPRINT ESC$;"[4m";Text$;ESC$;"[24m"¶
```

1.4. Astuces de travail avec Amiga

#21 : Toujours plus vite

La rapidité de traitement de votre Amiga sera accrue lors de longs calculs si vous fermez les écrans dont vous ne vous servez pas. Aidez votre processeur graphique et le système tout entier à travailler plus vite.

#22 : Stop !

Une pression sur le bouton gauche de la souris, interrompt le défilement d'un texte à l'écran. Une fois le bouton relâché, le défilement reprend. Voilà qui est bien utile lorsque le texte se déroule rapidement comme c'est le cas avec les commandes Dir, List et Type.

#23 : DiskDoctor Premiers Secours

Il vous est déjà probablement arrivé d'effacer un fichier par mégarde. Si vous agissez rapidement, vous pouvez récupérer le fichier. Amiga n'efface pas les fichiers à proprement parler. Il ne fait que redéfinir l'entrée correspondante dans le répertoire et les octets disponibles.

Point n'est besoin de posséder un gestionnaire de disque pour récupérer le fichier. Le programme Diskdoctor suffit. A condition que vous n'ayez pas écrit sur le disque entretemps.

#24 : Une méditation de moins

Lorsque vous lancez un programme à partir de Workbench, il s'exécute sans hésitation. Si vous le lancez à partir de CLI, il a besoin de 5 secondes de réflexion. Pourquoi ? de nombreux utilisateurs se posent cette question. Voici donc la réponse : sur un ordinateur multitâches, chaque programme possède sa propre pile où une adresse de retour est enregistrée. Lorsqu'un programme démarre à partir de Workbench, cette pile est créée à l'aide de la valeur contenue dans le fichier Info.

Les choses se présentent de façon différente dans CLI. Dans ce cas, c'est la valeur de la pile du CLI à la place de celle du fichier Info qui entre en action. Cette valeur est généralement assez petite. Nous vous rappelons, pour

mémoire, que lorsque vous lancez un programme à partir de CLI, celui ci accepte la taille de la pile CLI courante. Si la mémoire n'est pas suffisante, le Gourou va entrer en méditation. Pour remédier à cela, il faut agrandir la pile avant d'exécuter le programme. Pour connaître la taille de la pile, sélectionnez Info dans le menu Workbench.

#25 : Plusieurs exemplaires d'une même chose

Le fichier System-Configuration se trouve dans le répertoire devs: de n'importe quel disque d'initialisation. Ce fichier contient tous les paramètres de Preferences. Copiez ce fichier sur chacun de vos disques d'initialisation afin de retrouver les paramètres de Preferences sur tous les disques d'initialisation. Voici un exemple de commande CLI.

```
1> copy sys:devs/system-configuration df1:devs¶
```

1.5. Les cachettes des messages secrets

De nombreux composants logiciels et matériels sont agrémentés de messages par-ci, par-là : messages de bienvenue, nom des programmeurs, opinions politiques, histoires d'amour, etc... Ces messages dissimulés dans le système sont baptisés de plusieurs noms "Oeufs de Pâques", "Mémoire gaspillée", et autres. Nous vous en dévoilons quelques uns ci-dessous.

#26 : Message secret dans Kickstart

Les développeurs d'AMIGA sont commme toutes les personnes inventives. Si, dans un souci d'immortalité, ils voient une occasion d'aposer leur "griffe" sur les logiciels ou matériels qu'ils ont conçus , ils ne s'en privent pas. Vous trouverez, par exemple, gravée dans certains châssis, la signature de l'équipe qui a conçu la partie matérielle de l'Amiga.

La structure interne du matériel dissimule un message auquel vous pouvez accéder par l'intermédiaire de Workbench.

Ce programme en BASIC affiche le message caché lorsque vous entrez les adresses de départ et de fin nécessaires. Utilisez les valeurs correspondant à la version de Kickstart que vous possédez.

```
Adresses:¶
'V1.1 16653596 16653672 (Premiers Amiga 1000)¶
'V1.2 16649670 16649747 (Amiga 500 et 2000)¶
INPUT "Adresse de départ";Start¶
INPUT "Adresse de fin;Finish¶
PRINT¶
FOR i = Start to finish¶
 PRINT CHR$ (PEEK(i))¶
NEXT i¶
```

Nous vous encourageons à parcourir une partie de la mémoire en entrant d'autres valeurs. Vous aurez peut-être la chance de découvrir ce texte ou un texte similaire dans un autre emplacement. Si vous dénichez un nouveau message, faites-nous partager cette découverte, écrivez nous pour nous indiquer où il se cache.

#27 : Une histoire d'amour

Avez-vous déjà remarqué le texte secret, tapis dans le programme Preferences du Workbench 1.2 ? Une manipulation quelque peu compliquée vous laissera apercevoir un message inscrit dans la barre de titre de la fenêtre.

Affichez l'écran Preferences. Deux icônes symbolisant une souris sont dessinés sur la droite. Ces deux icônes possèdent deux boutons. Déplacez le pointeur sur le bouton gauche de la souris située à l'extrême gauche. Cliquez dessus puis déplacez le pointeur vers le bouton suivant. Cliquez dessus et déplacez le pointeur vers le bouton gauche de l'icône souris de droite. Cliquez sur ce bouton puis sur celui de droite. Répétez cette manipulation quatre fois de suite en cliquant chaque fois sur chaque bouton. Après un total de 16 clics, cliquez sur la case Change printer. Cliquez enfin sur la flèche vers le haut jusqu'à obtenir la première option. Cliquez une dernière fois sur la flèche pour visualiser l'histoire d'amour cachée dans la barre de titre.

#28 : Un message secret

Appuyez simultanément sur les touches <Shift> et <Alt> et sur une touche de fonction quelconque durant une impression. Ejectez en même temps une disquette de l'unité de lecture. Un autre message surgit sur l'écran.

Insérez à nouveau la disquette, vous obtiendrez soit le même message, soit un message différent. Seules les anciennes versions affichent deux textes différents. Dans les nouvelles versions, pour visualiser ce deuxième texte, vous devez insérer une disquette dont l'icône se trouve déjà sur l'écran Workbench. Vous aurez assez de temps pour le faire.

Chapitre 2

Trucs et astuces de programmation

en AmigaBasic

Ce chapitre s'adresse aux programmeurs qui désirent faire de leurs programmes des modèles de convivialité. L'époque de l'ordinateur personnel à l'aspect rébarbatif est depuis longtemps révolue. Les premiers ordinateurs devaient être rebranchés chaque fois que l'on voulait exécuter un nouveau programme. Par la suite, les ordinateurs furent équipés de commutateurs qu'il fallait actionner pour emettre ou recevoir des données (les claviers sont apparus bien plus tard). Si, avant d'acquérir votre Amiga, vous avez été en possession d'un C64 ou de tout autre ordinateur Commodore, vous n'êtes pas sans vous rappeler les difficultés que rencontrait un débutant face au système d'exploitation (par exemple, taper LOAD"nom du fichier",8 uniquement pour charger un fichier à partir d'un disque).

L'orientation graphique de l'Amiga garantit la convivialité de chacun des aspects de l'interface utilisateur (choix des options, sélection et affichage). Pourquoi, dès lors, imposer au programmeur un environnement de programmation peu engageant ?

Différentes méthodes vous mèneront sur le chemin de la convivialité. La plupart de ces méthodes passent par Intuition et son interface graphique. Les accessoires graphiques (gadgets) que sont les icônes, les cadres de message, les zones de texte et de dialogue, guident les premiers pas d'un utilisateur novice et améliorent la productivité d'un utilisateur chevronné face à des applications utilisant ce type d'interface.

La majeure partie du présent chapitre est consacrée à l'optimisation de l'aspect convivial de vos programmes, l'accent étant mis sur l'entrée, la sélection et la gestion. Dans cette optique, la souris constitue le périphérique d'entrée privilégié.

2.1. Une fenêtre Intuition

Toutes les activités nécéssaires à la gestion d'un programme ne peuvent être réalisées à l'aide de menus. Il nous faut donc songer à des solutions de remplacement. Quelles sont ces solutions ? La disquette Workbench fournit quelques exemples. Preferencess constitue une excellente alternative aux menus déroulants. Ouvrez Preferences et vous voilà face à un écran convivial composé de gadgets booléens, proportionnels, de remplissage et même d'ascenceurs, qui vous permettent de sélectionner très simplement les différentes options qui s'offrent à vous.

Les gadgets proportionnels règlent les couleurs, la vitesse et le délai de répétition des touches ainsi que l'écart entre deux clics de la souris. Les gadgets de sélection proposent un nombre de caractères par ligne et le mode entrelacé. Les ascenceurs de Change Printer permettent à l'utilisateur de définir les paramètres d'impression. Quant aux gadgets booléens de l'écran principal, il suffit de cliquer dessus pour sauvegarder (Save), utiliser, (Use) et effacer (Cancel).

Les programmes suivants fournissent des exemples de construction de tous les accessoires graphiques (gadgets) que l'on vient d'énumérer. Pour visualiser ces accessoires, il faut avant tout ouvrir une fenêtre. Généralement, AmigaBASIC ouvre une fenêtre directement après l'avoir chargée. Les fenêtres BASIC présentant quelques restrictions, nous nous servirons de la bibliothèque Intuition pour ouvrir une fenêtre. Les bibliothèques du Système d'exploitation d'Amiga permettent, en effet, une gestion plus élaborée qu'une programmation en BASIC standard.

Le premier programme donné en exemple se limite à l'ouverture d' une fenêtre Intuition sur l'écran Workbench. Pour ce faire, nous appelons le fichier intuition.library, également utilisé dans les programmes illustrès dans la suite de ce chapitre. Les deux bibliothèques intuition.library et exec.library sont indispensables au fonctionnement de ce programme. Pour les créer, utilisez le programme BASIC ConvertFD situé dans le tiroir BASICDemos du disque Extras. Lorsque vous lancez le programme ConvertFD, assurez vous qu'aucun des paramètres suivants n'est omis ou erroné : disque, chemin, nom du fichier. Dans le cas contraire, le programme ConvertFD ne trouvera pas le fichier indiqué. Voici , à l'intention des détenteurs de Workbench 1.3, un exemple d'exécution de ConvertFD qui crée le fichier exec.bmap sur le RAM-disk:

```
Enter name of fd.file to read> "Extras
1.3:FD1.3/exec_lib.fd"¶
Enter name of bmap file to produce > ram:exec.bmap¶
```

Pour fonctionner, les programmes d'exemple ont besoin de exec.library (exec_lib.fd), intuition.library(intuition_lib.fd) et graphics.library (graphics_lib.fd).

Les fichiers bmaps sont rangés dans un tiroir intitulé bmaps sur le disque optionnel T&A2 qui accompagne le présent ouvrage. Par conséquent, les programmes d'exemple suivants recherchent les fichiers bmap dans ce tiroir. Si vos fichiers .bmaps résident sur un disque ou dans un tiroir différents, modifiez les commandes LIBRARY de façon à ce que les fichiers .bmap puissent être ouverts.

Le programme qui suit contient des lignes en BASIC qui s'étendent sur deux lignes. Les contraintes liées à la mise en page de ce livre nous ont obligés à diviser en deux des lignes de programme qui ,en réalité, ne doivent occuper qu'une seule ligne en Amiga BASIC. Dans ce cas la deuxième ligne n'a pas de marge. Prenons comme exemple la ligne de programme suivante :

```
 WinDef NWindow, 100, 50, 460, 150, 32+64+521&, ¶
15&+4096&,¶
0&, Title$(P)¶
```

Vous remarquerez qu'elle occupe deux lignes que la seconde ligne ne possède pas de marge. En AmigaBASIC, la totalité de la ligne doit tenir sur une ligne et une seule.

Ces précisions données, voici le programme qui ouvre une fenêtre Intuition sur un écran Workbench.

```
'*************************************¶
'*                                   *¶
'* Ouverture d'une fenêtre Intuition  *¶
'* ---------------------------------- *¶
'*                                   *¶
'* Auteur : Wolf-Gideon Bleek         *¶
'* Date   : 22 Mai 1988               *¶
'* Version: 1.1                      *¶
'* Système d'expl. :V1.2 & V1.3       *¶
'*                                   *¶
'*************************************¶
  OPTION BASE 1¶
  DEFLNG a-z¶
' Bmaps situés sur le disque intitulé T&A2, les
vôtres se¶
' trouvent peut être ailleurs.¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
¶
  MList      = 0&¶
¶
MainProgram:¶
  GOSUB OpenAll¶
¶
  ' Programme principal¶
  FOR i = 1 TO 10000 : NEXT i¶
¶
  GOSUB CloseAll¶
¶
END¶
¶
OpenAll:¶
  Title$  = "Ma première fenêtre Intuition"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, 0&, Title$¶
  WinBase = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
RETURN¶
CloseAll:¶
  CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
'-------------------------------------------¶
SUB DefChip(Buffer, Size) STATIC¶
 SHARED MList¶
  Size=Size+8¶
```

```
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur (attention, pas trop
large!)¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,gad      ' Premier cadre¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
```

Description du programme

Les éléments les plus importants, à savoir les trois sous-programmes à l'origine de l'ouverture de la fenêtre Intuition, se situent à la fin du programme. DefChip() demande un espace mémoire de la taille voulue. La fonction système AllocMem(), située dans exec.library, permet ,de réserver cet espace. La routine enregistre deux valeurs dans les huit

premiers octets (liste de la mémoire allouée). Undef() libère la mémoire affectée. Mlist libère l'ensemble des zones mémoire les unes après les autres.

Les deux premiers sous-programmes sont des routines d'initialisation. Le programme WinDef() détient, ici, la plus grande importance car il intègre les donnés spécifiées dans une structure NewWindow. Ces données sont indispensables à Intuition pour ouvrir une nouvelle fenêtre. Windef() se borne à placer les données dans une nouvelle zone mémoire. Le reste constitue un sous-programme du programme principal.

Les activités accomplies par les sous-programmes nous sont désormais connues. Procédons ,à présent, à l'analyse du programme principal. Ce dernier utilise des fonctions qui appartiennent aux bibliothèques Intuition et Exec; elles sont donc toutes les deux ouvertes. Le segment principal se branche ensuite sur la routine OpenAll. A cet endroit, un appel de la définition de la structure NewWindow à l'aide de Windef() est réalisé. Cette structure est transférée dans Intuition au moyen de OpenWindow(). Si la fenêtre est correctement ouverte, un pointeur est renvoyé sur la structure Window. En cas d'erreur, c'est la valeur 0 qui est renvoyée. La nouvelle structure contient toutes les données qui vous permettent de créer votre propre fenêtre Intuition.

Après l'exécution de OpenAll, le programme est interrompu par la boucle FOR/NEXT de façon à visualiser la fenêtre un instant. L'interpréteur BASIC se branche ensuite sur le sous-programme intitulé CloseAll qui a pour fonction de fermer la fenêtre et de libérer l'espace mémoire affecté.

Nous voici à présent en possession des éléments de base à partir desquels nous allons pouvoir contruire des programmes professionnels et conviviaux. La fenêtre Intuition va nous donner l'occasion d'intégrer les commodités offertes par le programme Preferencess. Examinons la première d'entre elles, les gadgets.

2.1.1. Les gadgets

Le gadget est l'un des accessoires graphiques du programme Preferencess. Si l'utilisateur positionne le curseur de la souris sur Change Printer, Edit Pointer ou Cancel et qu'il clique sur le bouton gauche, une action immédiate se produit.

Ces gadgets peuvent être programmés en BASIC. AmigaBASIC tire les données relatives aux gadgets employés, à partir d'Intuition. Les gadgets que nous allons construire pourront facilement être incorporés au programme de la fenêtre Intuition présenté plus haut. Le programme qui vient, définit un nouveau champ de gadget. Ce programme peut être sauvegardé indépendamment puis annexé plus tard ou bien être écrit directement à la suite du programme de la fenêtre Intuition :

```
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&        ' Longueur de la Structure
Gadget¶
  POKEL bs    ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%        ' Bord gauche¶
  POKEW bs+ 6,y%        ' Bord supérieur¶
  POKEW bs+ 8,b%        ' Largeur¶
  POKEW bs+10,h%        ' Hauteur¶
  POKEW bs+12,f%        ' Indicateurs¶
  POKEW bs+14,a%        ' Activation¶
  POKEW bs+16,T%        ' Type de Gadget¶
  POKEL bs+18,i         ' Présentation du Gadget¶
  POKEL bs+26,txt       '*Texte du Gadget¶
  POKEL bs+34,si        ' SpecialInfo¶
  POKEW bs+38,n%        ' ID du Gadget¶
END SUB¶
```

Cette routine place les valeurs d'un Gadget dans une zone mémoire. La variable bs contient l'adresse de base de la zone mémoire ainsi que la valeur de retour de la routine d'affectation de la mémoire. nx indique l'adresse initiale du champ libre suivant ,permettant ainsi d'utiliser plusieurs Gadgets. Nous nous servirons de cette valeur plus tard pour ajouter des Gadgets supplémentaires dans la fenêtre. x%, y%, b% représentent les dimensions du Gadget.

f% et a% représentent deux indicateurs (flags) dont nous reparlerons plus loin. t% définit le type du Gadget (dans cet exemple, sa valeur est égale à 1; nous ajouterons d'autres types de Gadgets par la suite). i et txt incluent des informations graphiques supplémentaires : i spécifie le type de contour et txt le texte qui remplira le gadget. Quant à n%, il renferme une valeur d'identification qui aidera la programme à reconnaître le Gadget sélectionné par l'utilisateur.

Insérez le programme GadgetDef ,ci-dessus, à la fin du programme de la fenêtre Intuition et entrez la ligne ci-dessous dans le sous-programme OpenAll:

```
GadgetDef Gadget,0&,50,50,90,15,0,1,1,0&,0&,1¶
```

L'appel à ce sous-programme définit notre nouveau Gadget. Le résultat produit est le suivant:

- L'adresse de la structure Gadget est située après la fermeture de la routine dans Gadget.
- La structure Gadget contient un seul élément
- La position est 50,50
- La largeur est de 90 pixels et la hauteur de 15 pixels
- Le gadget réagit à un simple clic
- Le gadget est de type booléen et ne peut être qu'activé.
- Le gadget n'a ni contour, ni texte (il est invisible)
- Aucune structure supplémentaire n'est nécessaire
- L'accès au gadget se fait par la valeur 1

La structure de gadget ainsi définie est reliée à la structure de fenêtre par la commande :

```
WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&4096&, Gadget, Title$(P)¶
```

Ne lancez pas le programme pour l'instant. Vous pourriez le faire mais nous vous le déconseillons. Le programme ne contient pas de routine de contrôle. Le Gadget est pour l'instant invisible puisque son contour n'a pas encore été défini. Si vous entrez Run maintenant, la fenêtre va apparaître et après quelques tatonnements avec la souris, peut-être pourrez vous localiser la zone affectée au gadget. Seulement, vous pouvez toujours cliquer, aucune réaction ne se produira. Il faut ,au préalable, écrire un autre sous-programme qui effectue un branchement à partir du programme principal lorsqu'il rencontre certaines informations.

Le segment de programme ci-dessous lit un gadget et recherche un nouveau message Intuition transmis par le port d'information de la fenêtre. Puis, il se branche sur un autre sous-programme ,IntuitionMsg, qui lit et interprète le résultat du message.

```
GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgram:¶
  GOSUB OpenAll¶
  ' Programme principal¶
  MainLoop:¶
   IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
   END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
```

La fonction GetMsg() permet de rechercher un message dans le port de message de la fenêtre. Ce port de message réside dans la structure de fenêtre que OpenWindow() nous renvoie. Pour lire le port de message, une nouvelle variable doit être initialisée dans cette routine:

```
UserPort = PEEKL(WinBase+86)¶
```

Nous pouvons à présent ajouter le sous-programme qui gère le message reçu. Cette routine détermine, en premier lieu, de quel type de message il s'agit. Les messages varient en fonction du type d'action entrepris sur la fenêtre. Nous sommes à même de manipuler le message CloseWindow. Ce message est émis lorsque l'utilisateur clique sur le gadget de fermeture de la fenêtre. Notre sous-programme affiche aussi à l'écran le numéro du Gadget écrit dans la structure de gadget.

```
IntuitionMsg:¶
  MsgTyp    = PEEKL(IntuiMsg+20)¶
  Item      = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
```

```
¶
  IF (MsgTyp = GADGETDOWN) THEN¶
     'Activé¶
     PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = GADGETUP) THEN¶
     'Mode rel-verify¶
     PRINT "UP   Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = CLOSEW) THEN¶
     'Gadget de fermeture de la fenêtre¶
     PRINT "CLOSEWINDOW"¶
     Info = 0¶
  END IF¶
RETURN¶
```

Assemblez maintenant les différents segments et exécutez le programme. La fenêtre Intuition apparaît sur l'écran Workbench. Il est temps d'effectuer un premier test. Cliquez dans la portion supérieure gauche de la fenêtre jusqu'à repérer le gadget encore invisible. Le gadget n'est pas délimité car la structure Gadget ne renferme pas les informations adéquates (le positionnement du gadget sur la fenêtre est 50,50). Lorsque vous cliquez dessus avec la souris, le gadget apparaît, relâchez la souris et il disparaît tandis que le numéro du gadget s'inscrit dans la fenêtre AmigaBASIC Output.

Vérifions maintenant le sous-programme de message Intuition. Cliquez sur le gadget de fermeture de la fenêtre. Dans un premier temps, le texte apparaît puis la fenêtre se ferme.
Le gros du travail est maintenant achevé.

Voici le programme complet :

```
'*********************************¶
'*                               *¶
'* Création d'un Gadget          *¶
'* avec Intuition                *¶
'* ----------------------------- *¶
'*                               *¶
'* Auteur : Wolf-Gideon Bleek    *¶
'* Date   : 23 Mai 1988          *¶
'* Nom    : Premier Gadget       *¶
'* Version: 1.2                  *¶
'* Système: V1.2 & V1.3          *¶
'*                               *¶
```

```
'**********************************¶
  OPTION BASE 1¶
  DEFLNG a-z¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgram:¶
  GOSUB OpenAll¶
  ' Main part¶
  MainLoop:¶
   IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
   END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
¶
OpenAll:¶
  GadgetDef Gadget, 0&, 50, 50, 90, 15, 0, 1, 1, 0&,
0&, 0&, 1¶
  Title$   = "Le gadget invisible"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, Gadget, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  UserPort = PEEKL(WinBase+86)¶
RETURN¶
¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
¶
IntuitionMsg:¶
  MsgTyp    = PEEKL(IntuiMsg+20)¶
  Item      = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
¶
```

```
  IF (MsgTyp = GADGETDOWN) THEN¶
     'activé¶
     PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = GADGETUP) THEN¶
     'rel verify mode¶
     PRINT "UP   Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = CLOSEW) THEN¶
     'Gadget de fermeture de la fenêtre¶
     PRINT "FERMETURE DE LA FENETRE"¶
     Info = 0¶
  END IF¶
RETURN¶
'-----------------------------------------¶
¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
```

```
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Inducateurs¶
  POKEL bs+18,gad      ' Premier Gadget¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&       ' Longueur de Gadget-Structure¶
  POKEL bs   ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%       ' Bord gauche¶
  POKEW bs+ 6,y%       ' Bord supérieur¶
  POKEW bs+ 8,b%       ' Largeur¶
  POKEW bs+10,h%       ' Hauteur¶
  POKEW bs+12,f%       ' Indicateurs¶
  POKEW bs+14,a%       ' Activation¶
  POKEW bs+16,T%       ' Type de Gadget¶
  POKEL bs+18,i        ' Présentation du Gadget¶
  POKEL bs+26,txt      '*Texte du Gadget¶
  POKEL bs+34,si       ' SpecialInfo¶
  POKEW bs+38,n%       ' ID du Gadget¶
END SUB¶
```

2.1.2. Contour et texte d'un Gadget

Il manque un élément essentiel à notre Gadget : sa visibilité. Jusque là, le gadget ne pouvait être localisé qu'à l'aveuglette et rendu apparent en cliquant sur le bouton de la souris.

Nous devons donc résoudre ce léger problème. Lors de la description du sous-programme GadgetDef, nous avons introduit les variables txt et i. GadgetDef utilise ces variables pour entrer les éléments graphiques de base d'Intuition. Nous pouvons définir une structure IntuiText à l'aide de la variable txt et délimiter un contour à l'aide de la variable i.

Occupons-nous d'abord du texte. Nous avons besoin pour cela de la structure IntuiText. Celle-ci définit l'emplacement, la couleur, la police de caractères, la taille des caractères et enfin le texte proprement dit. Le sous-programme suivant attribue une zone mémoire et y place les données nécessaires:

```
SUB IntuiText(bs, cl%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1   'Long. Struct+ Long. texte+
Octet nul¶
  DefChip bs,Size¶
  POKE  bs    ,cl%      'FrontPen¶
  POKE  bs+  2,1        'Mode graphique¶
  POKEW bs+  4,x%       'Angle gauche¶
  POKEW bs+  6,y%       'Angle supérieur¶
  POKEL bs+12,bs+20     'Itexte¶
  POKEL bs+16,nx        'Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
```

A la suite de l'adresse de départ de la structure, nous insérons la couleur des caractères et le point de départ du texte exprimé en pixels. Puisque le pointeur accepte un texte plus long, il est possible de fournir un autre pointeur IntuiText en tant que valeur finale.

La routine sélectionne JAM2 comme mode graphique (c'est à dire que les couleurs de premier plan et de fond sont "amalgamées" dans la zone sélectionnée). Ainsi, le mode graphique se superpose sur le fond de manière à laisser apparaître un texte au contour net. La seconde couleur peut être ajustée par l'ajout d'une valeur dans la liste de paramètres et par l'écriture (POKE) de la valeur de la nouvelle couleur dans bs+1 !. Pour insérer le texte à l'intérieur du Gadget, vous devez spécifier ce texte dans le sous-programme puis l'ajouter à la routine de définition Gadget comme suit :

```
TestTxt$ = "Test-Text"¶
IntuiText Text, 2, 10, 2, TestTxt$, 0&¶
GadgetDef Gadget, 0&, 50, 50, 90, 15, 0, 1, 1, Edge,
Text,0&, 1¶
```

L'étape suivante concerne le contour du Gadget. L'objectif est de construire un contour à l'intérieur duquel on pourra cliquer à l'aide de la souris. Ces lignes de démarcation nous permettront, en outre, de souligner un texte si nécessaire. Les bordures étant également transférées dans la structure de gadget, nous pouvons créer une structure distincte. Cette dernière necéssite un tableau de coordonnées, la couleur et la position qui pourront être traitées comme une structure de contour normale. Les coordonnées apparaissent en mémoire à la suite de la structure.

Nous avons mis au point un programme quelque peu différent pour la structure de contour. Ce programme introduit la valeur en mémoire et calcule les valeurs des coordonnées necéssaires au tableau. De la sorte, la définition d'un encadré pour le Gadget de message est extrêmement simple. Voici la fonction dans son intégralité :

```
SUB Border(bs, x%, y%, c%, b%, h%) STATIC¶
  DefChip  bs,48&      ' Long. Structure + tableau de
coord.¶
  POKEW  bs  ,x%       ' Angle gauche¶
  POKEW  bs+2,y%       ' Angle supérieur¶
  POKE   bs+4,c%       ' FrontPen¶
  POKE   bs+7,8        ' Compteur¶
  POKEL  bs+8,bs+16    ' *XY¶
  FOR i%=0 TO 1¶
     POKEW  bs+22+i%*4,h%-1¶
     POKEW  bs+24+i%*4,b%-1¶
     POKEW  bs+32+i%*4,1¶
     POKEW  bs+38+i%*4,h%-1¶
     POKEW  bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
```

La routine ci-dessus contient les valeurs nécessaires à la définition du contour. Nous insérons ensuite l'adresse de base de la structure dans la définition du gadget, de la même façon que pour la structure IntuiText. A présent, la structure Border est combinée à la structure Gadget. Voici un exemple:

```
Border Edge, 0, 0, 3, 90, 15¶
GadgetDef Gadget,0&, 50, 50, 90, 15, 0, 1,
1,Edge,Text,0&,1¶
```

Le Gadget est maintenant muni d'un texte et encadré d'une bordure. Vous pouvez l'agrandir si vous le désirez.

Nous vous donnons ci-dessous le programme complet, composé des sous-programmes et définitions décrits précédemment. Vous pouvez vous y référer pour vérifier si vous avez correctement agencé les différents modules.

```
'*********************************¶
'*                               *¶
'* Création d'un Gadget          *¶
'* avec Intuition                *¶
```

```
'* ------------------------------ *¶
'*                                *¶
'* Auteur : Wolf-Gideon Bleek     *¶
'* Date   : 23 Mai 1988           *¶
'* Nom    : Gadgets               *¶
'* Version: 1.2                   *¶
'* Système: V1.2 & V1.3           *¶
'*                                *¶
'**********************************¶
¶
 OPTION BASE 1¶
  DEFLNG a-z¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgram:¶
  GOSUB OpenAll¶
  ' Main part¶
  MainLoop:¶
   IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
   END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
¶
OpenAll:¶
  Border Edge, 0, 0, 3, 90, 15¶
  TestTxt$ = "Test-Text"¶
  IntuiText Text, 2, 10, 2, TestTxt$, 0&¶
  GadgetDef Gadget, 0&, 50, 50, 90, 15, 0, 1, 1,
Edge, Text, 0&, 1¶
  Title$ = "Mon premier Gadget complet"¶
  WinDef NWindow, 100, 100, 460, 100, 32+64+512&,
15&+4096&,Gadget, Title$¶
  WinBase   = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
```

```
RETURN¶
¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
¶
IntuitionMsg:¶
  MsgTyp     = PEEKL(IntuiMsg+20)¶
  Item       = PEEKL(IntuiMsg+28)¶
  GadgetNr%  = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
¶
  IF (MsgTyp = GADGETDOWN) THEN¶
     'activé¶
     PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = GADGETUP) THEN¶
     'rel verify mode¶
     PRINT "UP   Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = CLOSEW) THEN¶
     'Gadget de ferrmeture de la fenêtre¶
     PRINT "FERMETURE DE LA FENETRE"¶
     Info = 0¶
  END IF¶
RETURN¶
'-------------------------------------------¶
¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList>0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
```

```
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,gad      ' Premier Gadget¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&       ' Longueur de Gadget-Structure¶
  POKEL bs   ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%       ' Bord gauche¶
  POKEW bs+ 6,y%       ' Bord supérieur¶
  POKEW bs+ 8,b%       ' Largeur¶
  POKEW bs+10,h%       ' Hauteur¶
  POKEW bs+12,f%       ' Indicateurs¶
  POKEW bs+14,a%       ' Activation¶
  POKEW bs+16,T%       ' Type de Gadget¶
  POKEL bs+18,i        ' Présentation du Gadget¶
  POKEL bs+26,txt      '*Texte du Gadget¶
  POKEL bs+34,si       ' SpecialInfo¶
  POKEW bs+38,n%       ' ID du Gadget¶
END SUB¶
¶
SUB IntuiText(bs, cl%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1    'Long. Struct+ Long. texte+
Octet nul¶
  DefChip bs,Size¶
  POKE  bs    ,cl%     'FrontPen¶
  POKE  bs+  2,1       'Mode graphique¶
  POKEW bs+  4,x%      'Angle gauche¶
  POKEW bs+  6,y%      'Angle supérieur¶
  POKEL bs+12,bs+20    'Itexte¶
  POKEL bs+16,nx       'Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
```

```
      POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
¶
SUB Border(bs, x%, y%, c%, b%, h%) STATIC¶
  DefChip  bs,48&      ' Long. Structure + tableau de
coord.¶
¶
  POKEW  bs  ,x%       ' Angle gauche¶
  POKEW  bs+2,y%       ' Angle supérieur¶
  POKE   bs+4,c%       ' FrontPen¶
  POKE   bs+7,8        ' Compteur¶
  POKEL  bs+8,bs+16    ' *XY¶
  FOR i%=0 TO 1¶
     POKEW  bs+22+i%*4,h%-1¶
     POKEW  bs+24+i%*4,b%-1¶
     POKEW  bs+32+i%*4,1¶
     POKEW  bs+38+i%*4,h%-1¶
     POKEW  bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
```

2.1.3. Des Gadgets familiers

Nous vous proposons ,à présent, de définir quatre gadgets qui sont employés dans de nombreux programmes. Ces gadgets que vous connaissez bien sont OK, Cancel, Reset et Undo. Cliquer sur OK permettrait de poursuivre l'exécution d'un programme, Cancel d'interrompre une fonction , tandis que Reset pourrait être utilisé pour restaurer des variables à leur état initial et Undo pour redéfinir la dernière variable modifiée.

Ces différentes opérations sont réalisées selon un ordre précis: d'abord la définition du gadget, du texte et du contour, puis nous dresserons la liste des sous-programmes de lecture avec des sous-programmes vides. La sélection de l'un des quatre Gadgets de message appelle l'un des sous-programmes.

Définitions des gadgets :

```
  Border Bord, -1, -1, 1,67,14¶
  IntuiText OKTxt, 1, 26, 2, "OK", 0&¶
  IntuiText CancelTxt, 1, 10, 2, "Cancel", 0&¶
  IntuiText ResetTxt, 1, 14, 2, "Reset", 0&¶
  IntuiText UndoTxt, 1, 20, 2, "Undo", 0&¶
  GadgetDef UndoGad, 0&, 380, 52, 65, 12, 0, 1, 1,
```

```
Bord, UndoTxt, 0&, 1¶
  GadgetDef ResetGad, UndoGad, 380, 68, 65, 12, 0,
1, 1, Bord, ResetTxt, 0&, 2¶
  GadgetDef OKGad, ResetGad, 380, 84, 65, 12, 0, 1,
1, Bord, OKTxt, 0&, 3¶
  GadgetDef CancGad, OKGad, 380, 100, 65, 12, 0, 1,
1, Bord, CancelTxt, 0&, 4¶
  Title$ = "Exemple de quatre Gadgets de message
familiers"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, CancGad, Title$¶
```

Sous-programmes de lecture :

```
'Routines de vérification:¶
IF (MsgType = GADGETDOWN) THEN¶
 'activation¶
 PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
END IF¶
¶
IF (MsgType = GADGETUP) THEN¶
 'rel verify mode¶
 PRINT "UP Gadget-Nr.:";GadgetNr%¶
 IF GadgetNr% = 1 THEN¶
  GOSUB Undo   ' place l'ancienne valeur¶
 END IF¶
 IF GadgetNr% = 2 THEN¶
  GOSUB ResetRoutine  'ens. des valeurs à leur état
initial¶
 END IF¶
 IF GadgetNr% = 3 THEN¶
  GOSUB Ok  ' entrée de la valeur de fin¶
 END IF¶
 IF GadgetNr% = 4 THEN¶
  GOSUB Cancel  ' entrée de la valeur d'interruption¶
 END IF¶
END IF¶
¶
IF (MsgType = CLOSEW) THEN¶
 'Gadget de fermeture de la fenêtre¶
 PRINT "FERMETURE DE LA FENETRE"¶
 Info = 0¶
END IF¶
¶
RETURN¶
¶
Undo:¶
   PRINT "Le Gadget UNDO a été sélectionné"¶
RETURN¶
¶
ResetRoutine:¶
```

```
   PRINT "Le Gadget RESET a été sélectionné"¶
RETURN¶
¶
Ok:¶
   PRINT "Le Gadget OK a été sélectionné"¶
RETURN¶
¶
Cancel:¶
   PRINT "Le Gadget CANCEL a été sélectionné"¶
RETURN¶
```

Listing complet du programme avec intégration des nouvelles définitions de gadget :

```
'**********************************¶
'*                                 *¶
'* Des Gadgets familiers           *¶
'* ------------------------------- *¶
'*                                 *¶
'* Auteur : Wolf-Gideon Bleek      *¶
'* Date   : 23 Mai 1988            *¶
'* Nom    : Gadgets familiers      *¶
'* Version: 1.2                    *¶
'* Système: V1.2 & V1.3            *¶
'*                                 *¶
'**********************************¶
  OPTION BASE 1¶
  DEFLNG a-z¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgram:¶
  GOSUB OpenAll¶
  ' Mani part¶
  MainLoop:¶
    IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
    END IF¶
¶
```

```
  GOSUB CloseAll¶
¶
END¶
¶
OpenAll:¶
¶
  Border Bord, -1, -1, 1,67,14¶
  IntuiText OKTxt, 1, 26, 2, "OK", 0&¶
  IntuiText CancelTxt, 1, 10, 2, "Cancel", 0&¶
  IntuiText ResetTxt, 1, 14, 2, "Reset", 0&¶
  IntuiText UndoTxt, 1, 20, 2, "Undo", 0&¶
  GadgetDef UndoGad, 0&, 380, 52, 65, 12, 0, 1, 1,
Bord, UndoTxt, 0&, 1¶
  GadgetDef ResetGad, UndoGad, 380, 68, 65, 12, 0,
1,
1, Bord, ResetTxt, 0&, 2¶
  GadgetDef OKGad, ResetGad, 380, 84, 65, 12, 0, 1,
1, Bord, OKTxt, 0&, 3¶
  GadgetDef CancGad, OKGad, 380, 100, 65, 12, 0, 1,
1, Bord, CancelTxt, 0&, 4¶
  Title$ = "Exemple de quatre Gadgets de message
familiers"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, CancGad, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
RETURN¶
¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
¶
IntuitionMsg:¶
  MsgType   = PEEKL(IntuiMsg+20)¶
  Item      = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
'The check routines:¶
IF (MsgType = GADGETDOWN) THEN¶
 'activation¶
 PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
END IF¶
¶
IF (MsgType = GADGETUP) THEN¶
 'rel verify mode¶
 PRINT "UP Gadget-Nr.:";GadgetNr%¶
 IF GadgetNr% = 1 THEN¶
  GOSUB Undo    ' place l'ancienne valeur¶
 END IF¶
 IF GadgetNr% = 2 THEN¶
```

```
  GOSUB ResetRoutine  'ens. des valeurs à leur état
initial¶
 END IF¶
 IF GadgetNr% = 3 THEN¶
  GOSUB Ok  ' entrée de la valeur de fin¶
 END IF¶
 IF GadgetNr% = 4 THEN¶
  GOSUB Cancel  ' entrée de la valeur
d''interruption¶
 END IF¶
END IF¶
¶
IF (MsgType = CLOSEW) THEN¶
 'Gadget de fermeture de la fenêtre¶
 PRINT "CLOSE WINDOW"¶
 Info = 0¶
END IF¶
¶
RETURN¶
¶
Undo:¶
   PRINT "Le Gadget UNDO a été sélectionné"¶
RETURN¶
¶
ResetRoutine:¶
   PRINT "Le Gadget RESET a été sélectionné"¶
RETURN¶
¶
Ok:¶
   PRINT "Le Gadget OK a été sélectionné"¶
RETURN¶
¶
Cancel:¶
   PRINT "Le Gadget CANCEL a été sélectionné"¶
RETURN¶
¶
'-------------------------------------------¶
¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
SUB UnDef STATIC¶
 SHARED MList¶
```

```
undef.loop:¶
  IF MList>0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,gad      ' Premier Gadget¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&       ' Longueur de Gadget-Structure¶
  POKEL bs   ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%       ' Bordure gauche¶
  POKEW bs+ 6,y%       ' Bordure supérieure¶
  POKEW bs+ 8,b%       ' Largeur¶
  POKEW bs+10,h%       ' Hauteur¶
  POKEW bs+12,f%       ' Indicateurs¶
  POKEW bs+14,a%       ' Activation¶
  POKEW bs+16,T%       ' Type du Gadget¶
  POKEL bs+18,i        ' Présentation du Gadget¶
  POKEL bs+26,txt      '*Texte du Gadget¶
  POKEL bs+34,si       ' SpecialInfo¶
  POKEW bs+38,n%       ' GadgetID¶
END SUB¶
¶
SUB IntuiText(bs, c1%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1    ' Long. de structure+ Long
Texte+ Octet nul¶
  DefChip bs,Size¶
  POKE  bs   ,c1%      ' FrontPen¶
  POKE  bs+ 2,1        ' DrawMode¶
  POKEW bs+ 4,x%       ' Angle gauche¶
  POKEW bs+ 6,y%       ' Angle supérieur¶
```

```
  POKEL bs+12,bs+20    ' ITexte¶
  POKEL bs+16,nx       ' Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
¶
SUB Border(bs, x%, y%, c%, b%, h%) STATIC¶
  DefChip bs,48&       ' Long. Structure+ Tableau de
coord.¶
  POKEW bs  ,x%        ' Angle gauche¶
  POKEW bs+2,y%        ' Angle supérieur¶
  POKE  bs+4,c%        ' FrontPen¶
  POKE  bs+7,8         ' Compteur¶
  POKEL bs+8,bs+16     '*XY¶
  FOR i%=0 TO 1¶
     POKEW bs+22+i%*4,h%-1¶
     POKEW bs+24+i%*4,b%-1¶
     POKEW bs+32+i%*4,1¶
     POKEW bs+38+i%*4,h%-1¶
     POKEW bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
```

2.1.4. Les Gadgets de sélection

Lorsque vous créez de nouveaux Gadgets pour une fenêtre, vous devez prendre l'habitude de concevoir des ensembles de Gadgets liés à des actions spécifiques. Il n'est pas recommandé d'occuper une fenêtre avec des Gadgets isolés car, ce faisant, un seul type de sélection serait disponible. Prenons, à titre d'exemple, la sélection d'une police de caractères. Une seule police de caractères à la fois peut être activée. La nature même des Gadgets de sélection impose d'emblée des restrictions de choix. En effet, si vous cliquez sur une option pour la sélectionner, vous invalidez automatiquement l'option précédemment choisie (le nouveau Gadget sélectionné apparaît sur fond noir tandis que l'option précédemment choisie reprend un aspect normal).

Les programmes permettant d'élaborer des Gadgets de sélection diffèrent un peu des programmes dont il a été question précédemment (Gadgets de message). La création des structures, des graphiques et des textes demeure la même. Cette fois ci, cependant, la zone de fond du Gadget sélectionné doit changer de couleur. Intuition testera l'état du Gadget. Si c'est un Gadget de sélection qui est activé, il deviendra noir à la suite d'un clic. Pour noircir et blanchir le fond du Gadget, nous utilisons graphics.library. Ce fichier library comprend les commandes SetDrMd (SetDrawMode) et

RectFill(RectangleFill). Lorsque vous invoquez ce fichier, assurez vous de l'exactitude du chemin DOS. Dans l'exemple ci-dessous, on accède à graphics.library dans le répertoire bmaps sur le disque T&A2.

```
LIBRARY "T&A2:bmaps/graphics.library"¶
```

N'oubliez pas de renommer les chemins d'appel de LIBRARY pour les harmoniser avec vos propres répertoires.

Ne perdons pas de vue non plus que notre routine d'évaluation doit aussi vider de sa couleur (blanchir) un Gadget invalidé du fait d'une nouvelle sélection. Voici des exemples de Gadgets:

```
  Border Bord, -1, -1, 1, 100, 14¶
  IntuiText To60, 1, 10, 2, "Topaz 60", 0&¶
  IntuiText To80, 1, 10, 2, "Topaz 80", 0&¶
  IntuiText PC60, 1, 26, 2, "PC 60", 0&¶
  IntuiText PC80, 1, 26, 2, "PC 80", 0&¶
  GadgetDef To60Gad, 0&, 80, 61, 98, 12, 0, 1, 1,
Bord, To60, 0&, 1¶
  GadgetDef To80Gad, To60Gad, 80, 74, 98, 12, 0, 1,
1, Bord, To80, 0&, 2¶
  GadgetDef PC60Gad, To80Gad, 80, 87, 98, 12, 0, 1,
1, Bord, PC60, 0&, 3¶
  GadgetDef PC80Gad, PC60Gad, 80, 100, 98, 12, 0, 1,
1, Bord, PC80, 0&, 4¶
  Title$ = "Choix d'une police de caractères"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, PC80Gad, Title$¶
```

Tous les Gadgets utilisent la structure Border employée pour concevoir un tableau similaire à l'option "Format Papier" (Paper Size) du Programme Preferencess. Les quatre premiers textes du programme ci-dessus correspondent à quatre polices de caractères. Le programme crée ensuite un pointeur sur la liste de Gadgets et place cette liste dans la zone de définition de la fenêtre en même temps que son titre.

Après le démarrage du programme, nous visualisons dans la fenêtre un ensembles de Gadgets permettant de sélectionner une police de caractères. La sélection se fait en cliquant sur le Gadget sélectionné et se traduit par le noircissement de ce dernier. La routine de lecture d'un message Intuition ci-dessous accomplira ce changement d'aspect (fill.unfill).

```
IF (MsgTyp = GADGETUP) THEN¶
 'rel verify mode¶
 PRINT "UP Gadget-Nr.:";GadgetNr%¶
 IF active THEN¶
  SetDrMd RastPort, 2¶
  RectFill RastPort, 80&, 13*active+48&, 177&,
13*active+59&¶
  SetDrMd RastPort, 1¶
 END IF¶
 active = GadgetNr%¶
 SetDrMd RastPort, 2¶
 RectFill RastPort, 80&, 13*active+48&, 177&,
13*active+59&¶
 SetDrMd RastPort, 1¶
END IF¶
```

La routine de lecture fait en sorte que le programme reconnaisse les variables choisie par la police de caractère. Si une autre police est sélectionnée, son Gadget se noircit à son tour tandis que l'option précédemment choisie revient à l'état normal.

Nous analysons ensuite, les nouvelles polices de caractères choisies par l'utilisateur. Le programme calcule les coordonnées du contour et le déssine. Ensuite, la routine s'achève et revient au corps du programme.

Tous les tableaux peuvent être construits de façon analogue, quel que soit leur usage. Le programmeur doit prévoir le nombre de Gadgets et réaliser les calculs necéssaires à l'inversion du graphisme. Ceci ne devrait présenter aucune dificulté, les Gadgets étant tous de la même dimension et donc simple à calculer.

Voici le listing complet avec intégration des sous-programmes, des appels de bibliothèque et des variables.

```
'*********************************¶
'*                               *¶
'* Création de Gadgets de         *¶
'* sélection avec Intuition      *¶
'* ----------------------------- *¶
'*                               *¶
'* Auteur : Wolf-Gideon Bleek    *¶
'* Date   : May 23 '88           *¶
'* Name   : Fill-Gadgets         *¶
'* Version: 1.2                  *¶
'* Système : V1.2 & V1.3         *¶
'*                               *¶
'*********************************¶
```

```
  OPTION BASE 1¶
  DEFLNG a-z¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
  LIBRARY "T&A2:bmaps/graphics.library"¶
¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgramm:¶
  GOSUB OpenAll¶
  ' Mani part¶
  MainLoop:¶
    IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
    END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
¶
OpenAll:¶
  Border Bord, -1, -1, 1, 100, 14¶
  IntuiText To60, 1, 10, 2, "Topaz 60", 0&¶
  IntuiText To80, 1, 10, 2, "Topaz 80", 0&¶
  IntuiText PC60, 1, 26, 2, "PC 60", 0&¶
  IntuiText PC80, 1, 26, 2, "PC 80", 0&¶
  GadgetDef To60Gad, 0&, 80, 61, 98, 12, 0, 1, 1,
Bord, To60, 0&, 1¶
  GadgetDef To80Gad, To60Gad, 80, 74, 98, 12, 0, 1,
1, Bord, To80, 0&, 2¶
  GadgetDef PC60Gad, To80Gad, 80, 87, 98, 12, 0, 1,
1, Bord, PC60, 0&, 3¶
  GadgetDef PC80Gad, PC60Gad, 80, 100, 98, 12, 0, 1,
1, Bord, PC80, 0&, 4¶
  Title$ = "Choix de la police de caractères"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, PC80Gad, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
RETURN¶
```

```
¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
¶
IntuitionMsg:¶
  MsgTyp    = PEEKL(IntuiMsg+20)¶
  Item      = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
¶
  IF (MsgTyp = GADGETDOWN) THEN¶
     'activé immédiatement¶
     PRINT "DOWN Gadget-Nr.:";GadgetNr%¶ •
  END IF¶
¶
IF (MsgTyp = GADGETUP) THEN¶
 'rel verify mode¶
 PRINT "UP Gadget-Nr.:";GadgetNr%¶
 IF active THEN¶
  SetDrMd RastPort, 2¶
  RectFill RastPort, 80&, 13*active+48&, 177&,
13*active+59&¶
  SetDrMd RastPort, 1¶
 END IF¶
 active = GadgetNr%¶
 SetDrMd RastPort, 2¶
 RectFill RastPort, 80&, 13*active+48&, 177&,
13*active+59&¶
 SetDrMd RastPort, 1¶
END IF¶
¶
  IF (MsgTyp = CLOSEW) THEN¶
     'Gadget de fermeture de la fenêtre¶
     PRINT"FERMETURE DE LA FENETRE"¶
     Info = 0¶
  END IF¶
RETURN¶
'-----------------------------------------¶
¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
```

```
  END IF¶
END SUB¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList>0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,gad      ' Premier Gadget¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&       ' Longueur de gadget-structure¶
  POKEL bs   ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%       ' Bordure gauche¶
  POKEW bs+ 6,y%       ' Bordure supérieure¶
  POKEW bs+ 8,b%       ' Largeur¶
  POKEW bs+10,h%       ' Hauteur¶
  POKEW bs+12,f%       ' Indicateurs¶
  POKEW bs+14,a%       ' Activation¶
  POKEW bs+16,T%       ' Type du Gadget¶
  POKEL bs+18,i        ' Présentation du Gadget¶
  POKEL bs+26,txt      '*Texte du Gadget¶
  POKEL bs+34,si       ' SpecialInfo¶
  POKEW bs+38,n%       ' GadgetID¶
END SUB¶
¶
SUB IntuiText(bs, c1%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1    ' Long. Structure+ Longueur
Texte+ Octet nul¶
  DefChip bs,Size¶
```

```
  POKE  bs   ,c1%      ' FrontPen¶
  POKE  bs+ 2,1        ' DrawMode¶
  POKEW bs+ 4,x%       ' Angle gauche¶
  POKEW bs+ 6,y%       ' Angle supérieur¶
  POKEL bs+12,bs+20    ' ITexte¶
  POKEL bs+16,nx       ' Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
¶
SUB Border(bs, x%, y%, c%, b%, h%) STATIC¶
  DefChip bs,48&       ' Long. Structure + Tableau de
coord.¶
  POKEW bs  ,x%        ' Angle gauche¶
  POKEW bs+2,y%        ' Angle supérieur¶
  POKE  bs+4,c%        ' FrontPen¶
  POKE  bs+7,8         ' Compteur¶
  POKEL bs+8,bs+16     '*XY¶
  FOR i%=0 TO 1¶
     POKEW bs+22+i%*4,h%-1¶
     POKEW bs+24+i%*4,b%-1¶
     POKEW bs+32+i%*4,1¶
     POKEW bs+38+i%*4,h%-1¶
     POKEW bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
```

2.1.5. Les ascenceurs

Lorsque les options offertes par un écran deviennent pléthoriques, il s'avère difficile de les classer dans de simples Gadgets de sélection. Devant cette difficulté, ce sont logiquement, les ascenceurs qui s'imposent. Un ascenceur, à l'intérieur duquel on peut se déplacer vers le bas ou vers le haut, permet de ne visualiser qu'une partie des choix possibles.

Change Printer, dans le programme Preferencess, fournit un exemple d'ascenceur. En effet, les drivers d'imprimantes y sont présentés sous la forme d'un ascenceur dans l'angle supérieur droit de l'écran. L'intérêt de ce type d'ascenceurs est leur flexibilité. Peu importe le nombre des drivers d'imprimante figurant sur un disque, vous pouvez tous les visualiser et les sélectionner. Un clic sur les flèches situées à gauche de la liste des imprimantes entraine le défilement des options possibles.

Notre programme doit créer un tableau permettant de visualiser toutes les options offertes ainsi que deux Gadgets, un pour chaque flèche. A chaque fois que l'on clique sur une flèche, une nouvelle option apparaît. Le listing suivant définit notre tableau déroulant ainsi que deux Gadgets destinés aux flèches et trois zones d'affichage.

```
OpenAll:¶
  Border Bord, -1, -1, 1, 200, 14¶
  Border Box, 0, -1, 1, 50, 21¶
  GadgetDef Higher, 0&, 51, 60, 48, 18, 0, 1, 1, 0&,
0&, 0&, 1¶
  GadgetDef Lower, Higher, 51, 80, 48, 18, 0, 1, 1,
0&, 0&, 0&, 2¶
  Title$   = "Ascenceur"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, Lower, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
  DrawBorder RastPort, Bord, 100&, 60&¶
  DrawBorder RastPort, Bord, 100&, 73&¶
  DrawBorder RastPort, Bord, 100&, 86&¶
  DrawBorder RastPort, Box, 50&, 60&¶
  DrawBorder RastPort, Box, 50&, 79&¶
¶
  x = 50      : y = 60¶
 'x-            y- value¶
  x(1) =17+x : x(2) =16+y¶
  x(3) =34+x : x(4) =16+y¶
  x(5) =34+x : x(6) =10+y¶
  x(7) =40+x : x(8) =10+y¶
  x(9) =25+x : x(10)=2+y¶
  x(11)=10+x : x(12)=10+y¶
  x(13)=17+x : x(14)=10+y¶
  x(15)=17+x : x(16)=16+y¶
¶
  Move RastPort, 17+50&, 16+60&¶
  PolyDraw RastPort, 8&, VARPTR(x(1))¶
¶
  y = 62¶
 'y- Value¶
  x(2) =20+y¶
  x(4) =20+y¶
  x(6) =26+y¶
  x(8) =26+y¶
  x(10)=34+y¶
  x(12)=26+y¶
  x(14)=26+y¶
  x(16)=20+y¶
¶
```

```
  Move RastPort, 17+50&, 20+62&¶
  PolyDraw RastPort, 8&, VARPTR(x(1))¶
¶
  FOR i = 1 TO 5¶
     READ Table$(i)¶
     IntuiText ITxt(i), 1, 0, 0, Table$(i), 0&¶
  NEXT i¶
  TabOut Active¶
RETURN¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
```

Une fois la fenêtre ouverte et les Gadgets définis, DrawBorder() dessine les boîtes qui contiendront les options. PolyDraw() utilise graphics.library pour tracer un polygone. Cette fonction permet de créer des tableaux de coordonnées en seulement quelques lignes de programme. Nous utilisons ici la routine Polydraw() pour dessiner le graphisme des flèches directionnelles situées à gauche du tableau déroulant.

Ensuite, OpenAll lit cinq textes à partir des instructions DATA qui seront employées plus tard dans le tableau. La routine TabOut gère le défilement des options.

Voici le programme dans sa totalité avec la nouvelle routine TabOut et les instructions DATA:

```
'************************************¶
'*                                  *¶
'* Les ascenceurs                   *¶
'* ----------------------           *¶
'*                                  *¶
'* Auteur : Wolf-Gideon Bleek       *¶
'* Date   : May 31 '88              *¶
'* Name   : Scroll-Gadgets          *¶
'* Version: 1.2                     *¶
'* Système : V1.2 & V1.3            *¶
'*                                  *¶
'************************************¶
  OPTION BASE 1¶
  DEFLNG a-w¶
  DEFINT x¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
```

```
  LIBRARY "T&A2:bmaps/graphics.library"¶
¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
  Active     = 2¶
¶
  DIM x(16)¶
  DIM SHARED Table$(5), ITxt(5)¶
¶
MainProgramm:¶
  GOSUB OpenAll¶
  ' Main part¶
  MainLoop:¶
    IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
    END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
OpenAll:¶
  Border Bord, -1, -1, 1, 200, 14¶
  Border Box, 0, -1, 1, 50, 21¶
  GadgetDef Higher, 0&, 51, 60, 48, 18, 0, 1, 1, 0&,
0&, 0&, 1¶
  GadgetDef Lower, Higher, 51, 80, 48, 18, 0, 1, 1,
0&, 0&, 0&, 2¶
  Title$   = "Ascenceur"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, Lower, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
  DrawBorder RastPort, Bord, 100&, 60&¶
  DrawBorder RastPort, Bord, 100&, 73&¶
  DrawBorder RastPort, Bord, 100&, 86&¶
  DrawBorder RastPort, Box, 50&, 60&¶
  DrawBorder RastPort, Box, 50&, 79&¶
¶
  x = 50      : y = 60¶
 'x-            y- value¶
  x(1) =17+x : x(2) =16+y¶
  x(3) =34+x : x(4) =16+y¶
  x(5) =34+x : x(6) =10+y¶
  x(7) =40+x : x(8) =10+y¶
```

```
  x(9) =25+x : x(10)=2+y¶
  x(11)=10+x : x(12)=10+y¶
  x(13)=17+x : x(14)=10+y¶
  x(15)=17+x : x(16)=16+y¶
¶
  Move RastPort, 17+50&, 16+60&¶
  PolyDraw RastPort, 8&, VARPTR(x(1))¶
¶
  y = 62¶
 'y- Value¶
  x(2) =20+y¶
  x(4) =20+y¶
  x(6) =26+y¶
  x(8) =26+y¶
  x(10)=34+y¶
  x(12)=26+y¶
  x(14)=26+y¶
  x(16)=20+y¶
¶
  Move RastPort, 17+50&, 20+62&¶
  PolyDraw RastPort, 8&, VARPTR(x(1))¶
¶
  FOR i = 1 TO 5¶
     READ Table$(i)¶
     IntuiText ITxt(i), 1, 0, 0, Table$(i), 0&¶
  NEXT i¶
  TabOut Active¶
RETURN¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
IntuitionMsg:¶
  MsgTyp     = PEEKL(IntuiMsg+20)¶
  Item       = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
¶
  IF (MsgTyp = GADGETDOWN) THEN¶
     'activé¶
     PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = GADGETUP) THEN¶
     'mode test¶
     PRINT "UP   Gadget-Nr.:";GadgetNr%¶
     IF GadgetNr% = 1 AND Active<>4 THEN Active=
Active+1 : TabOut(Active)¶
     IF GadgetNr% = 2 AND Active<>1 THEN Active=
Active-1 : TabOut(Active)¶
  END IF¶
¶
```

```
  IF (MsgTyp = CLOSEW) THEN¶
     'Gadget de fermeture de la fenêtre¶
     PRINT "CLOSE WINDOW"¶
     Info = 0¶
  END IF¶
RETURN¶
'-----------------------------------------¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList>0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, Gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Bordure gauche¶
  POKEW bs+ 2,y%       ' Bordure supérieure¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,Gad      ' Premier Gadget¶
  POKEL bs+26,bs+48    ' Titre¶
  POKEW bs+46,1        ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
txt, si, n%) STATIC¶
  DefChip bs,44&       ' Longueur de Gadget-Structure¶
```

```
  POKEL bs   ,nx       '*Gadget suivant¶
  POKEW bs+ 4,x%       ' Bordure gauche¶
  POKEW bs+ 6,y%       ' Bordure supérieure¶
  POKEW bs+ 8,b%       ' Largeur¶
  POKEW bs+10,h%       ' Hauteur¶
  POKEW bs+12,f%       ' Indicateurs¶
  POKEW bs+14,a%       ' Activation¶
  POKEW bs+16,T%       ' Type du Gadget¶
  POKEL bs+18,i        ' Présentation du Gadget¶
  POKEL bs+26,txt      '*Texte du Gadget¶
  POKEL bs+34,si       ' SpecialInfo¶
  POKEW bs+38,n%       ' GadgetID¶
END SUB¶
SUB IntuiText(bs, c1%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1    ' Long struct + Long. Texte+
Octet nul¶
  DefChip bs,Size¶
  POKE  bs   ,c1%      ' FrontPen¶
  POKE  bs+ 2,1        ' DrawMode¶
  POKEW bs+ 4,x%       ' Bordure gauche¶
  POKEW bs+ 6,y%       ' Bordure supérieure¶
  POKEL bs+12,bs+20    ' ITexte¶
  POKEL bs+16,nx       ' Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB Border(bs, x%, y%, C%, b%, h%) STATIC¶
  DefChip bs,48&       ' Long. structure + tableau
coord.¶
  POKEW bs  ,x%        ' Bordure gauche¶
  POKEW bs+2,y%        ' Bordure supérieure¶
  POKE  bs+4,C%        ' FrontPen¶
  POKE  bs+7,8         ' Compteur¶
  POKEL bs+8,bs+16     '*XY¶
  FOR i%=0 TO 1¶
     POKEW bs+22+i%*4,h%-1¶
     POKEW bs+24+i%*4,b%-1¶
     POKEW bs+32+i%*4,1¶
     POKEW bs+38+i%*4,h%-1¶
     POKEW bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
'exchange returned in PropInfo¶
SUB STRINGINFO(bs,max%,buff$) STATIC¶
  IF LEN(buff$)>max% THEN¶
     nmax%=LEN(buff$)¶
  ELSE¶
     nmax%=max%¶
  END IF¶
  IF (nmax% AND 1) THEN nmax%=nmax%+1¶
  Size=36+2*(nmax%+4)¶
```

```
  DefChip bs,Size¶
  POKEL bs,bs+36¶
  POKEL bs+4,bs+40+nmax%¶
  POKEW bs+10,max%+1¶
  IF buff$<>""THEN¶
    FOR i%=1 TO LEN(buff$)¶
      POKE bs+35+i%,ASC(MID$(buff$,i%,1))¶
    NEXT¶
  END IF¶
END SUB¶
SUB TabOut(Active) STATIC¶
SHARED RastPort¶
   COLOR 0,0¶
   FOR i = 0 TO 2¶
     SetAPen RastPort, 0¶
     RectFill RastPort, 101&, 13*i+60&, 296&, 13*i+
71&¶
   NEXT i¶
   COLOR 1,0¶
   FOR i = Active-1 TO Active+1¶
      IF i>0 AND i<5 THEN¶
         POKEW ITxt(i)+6, 62+(i-Active+1)*13¶
         PrintIText RastPort, ITxt(i), 110&, 0&¶
      END IF¶
   NEXT i¶
   SetDrMd RastPort, 2¶
   RectFill RastPort, 101&, 73&, 296&, 84&¶
   SetDrMd RastPort, 1¶
END SUB¶
DATA Scroll-Table¶
DATA Closer¶
DATA Table¶
DATA Gadget¶
DATA System gadgets¶
```

2.1.6. Les Gadgets proportionnels

Les sliders permettent à l'utilisateur de sélectionner des valeurs qui varient de façon proportionnelle. La définition de la couleur de l'écran dans le programme Preferences est un excellent exemple de Gadget proportionnel. Chaque contrôleur accepte une valeur comprise entre 0 et 15 représentant une intensité lumineuse qui peut aller de nulle (O) à forte (15).

Vous pourriez ,aussi bien, ajouter à l'écran trois zones de texte et y spécifier une valeur représentant l'intensité désirée. Les Gadget proportionnels sont tout de même plus pratiques pour ce genre de manipulation. Vous pouvez modifier l'intensité des couleurs de l'écran (Rouge, Vert et Bleu) en faisant glisser la boule vers la gauche ou vers la droite à l'aide de la souris.

La fichier intuition.library aide à programmer les Gadget proportionnels. Un Gadget proportionnel est composé d'une baguette le long de laquelle une boule se déplace. La boule se dirige verticalement, horizontalement ou dans les deux sens. Deux choix s'offrent à l'utilisateur : définir le graphisme du bouton-glisseur ou utiliser le graphisme par défaut fournit par Intuition.Le contour de la baguette est visible pour faciliter les manipulations.

La structure Gadget doit être agrandie de façon à contenir la structure PropInfo. Cette dernière est liée au pointeur SpecialInfo. Nous disposons, pour cela, d'un sous-programme qui place les paramètres en mémoire.

```
SUB PropInfo(bs, Flags%, HPot%, VPot%, HBody%,
VBody%) STATIC¶
  DefChip bs,22&¶
  POKEW bs   ,Flags%¶
  POKEW bs+ 2,HPot%¶
  POKEW bs+ 4,VPot%¶
  POKEW bs+ 6,HBody%¶
  POKEW bs+ 8,VBody%¶
END SUB¶
```

Nous devons accéder d'abord aux valeurs ajoutées à la structure Info. Des drapeaux (flags) indiquent le sens de déplacement de la boule : horizontalement (2) ou verticalement (4). Le drapeau autoknob informe Intuition qu'aucun graphisme n'a été défini pour la boule et qu'il faut par conséquent employer le graphisme par défaut fournit par Intuition. HPot et VPot spécifient la position de la boule. 0 indique l'axe inférieur droit tandis que &HFFFF indique l'axe supérieur gauche. Après avoir déplacé la boule, l'utilisateur peut lire sa nouvelle position à partir de cet endroit. HBody et VBody renvoient le pas de la boule. Ces deux valeurs sont calculées comme partie du tout (&HFFFF). Intuition insère toutes les autres valeurs rencontrées dans la structure- ces valeurs ne doivent pas être définies par le programme.

Autoknob définit le graphisme de la boule. Autoknob necéssite une zone mémoire de huit octets contenant la position (coordonnées X et Y) et la largeur de la boule. Si ces valeurs ne sont pas définies, la routine d'initialisation s'en chargera. Nous avons encore besoin de deux structures supplémentaires pour compléter nos Gadget proportionnels.

```
PropInfo PropI, 1+2, 0, 0, &HFFF, 0¶
IntuiText, Text, 2, -80, 2, "Mover:", 0&¶
DefChip Buffer, 8&¶
GadgetDef Gadget, 0&, 150, 30, 100, 10, 0, 1+2, 3,
Buffer, Text, PropI, 1¶
```

Nous pouvons construire des Gadget proportionnels complets à partir de ces quelques données. Nous vous présentons, à titre d'exemple, un programme qui crée trois gadgets de ce type. Ces Gadget proportionnels permettent de modifier la valeur liée à l'intensité d'une couleur à l'aide de la souris.

```
'************************************¶
'*                                  *¶
'* Gadgets proportionnels           *¶
'* ----------------------           *¶
'*                                  *¶
'* Auteur : Wolf-Gideon Bleek       *¶
'* Date   :May 23 '88               *¶
'* Name   :Proportional-Gadgets     *¶
'* Version: 1.2                     *¶
'* Système: V1.2 & V1.3             *¶
'*                                  *¶
'************************************¶
  OPTION BASE 1¶
  DEFLNG a-z¶
¶
  LIBRARY "T&A2:bmaps/exec.library"¶
  DECLARE FUNCTION AllocMem LIBRARY¶
  DECLARE FUNCTION GetMsg LIBRARY¶
  LIBRARY "T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION OpenWindow LIBRARY¶
  GADGETDOWN = 32&¶
  GADGETUP   = 64&¶
  CLOSEW     = 512&¶
¶
  MList      = 0&¶
  Info       = 1¶
¶
MainProgramm:¶
  GOSUB OpenAll¶
  ' Main part¶
```

```
  MainLoop:¶
    IF Info = 1 THEN¶
      IntuiMsg = GetMsg(UserPort)¶
      IF IntuiMsg > 0 THEN GOSUB IntuitionMsg¶
      GOTO MainLoop¶
    END IF¶
¶
  GOSUB CloseAll¶
¶
END¶
OpenAll:¶
  IntuiText RedTxt, 2, -80, 2, "Red", 0&¶
  IntuiText GrnTxt, 2, -80, 2, "Green", 0&¶
  IntuiText BluTxt, 2, -80, 2, "Blue", 0&¶
  PropInfo Prop1, 1+2, 0, 0, &HFFF, 0¶
  PropInfo Prop2, 1+2, 0, 0, &HFFF, 0¶
  PropInfo Prop3, 1+2, 0, 0, &HFFF, 0¶
  DefChip Buffer(1), 8&¶
  DefChip Buffer(2), 8&¶
  DefChip Buffer(3), 8&¶
  GadgetDef RedGad, 0&, 150, 30, 114, 10, 0, 1+2, 3,
Buffer(1), RedTxt, Prop1, 1¶
  GadgetDef GrnGad, RedGad, 150, 45, 114, 10, 0,
1+2, 3, Buffer(2), GrnTxt, Prop2, 2¶
  GadgetDef BluGad, GrnGad, 150, 60, 114, 10, 0,
1+2, 3, Buffer(3), BluTxt, Prop3, 3¶
  Title$   = "Color initialization"¶
  WinDef NWindow, 100, 50, 460, 150, 32+64+512&,
15&+4096&, BluGad, Title$¶
  WinBase  = OpenWindow(NWindow)¶
  IF WinBase = 0 THEN ERROR 7¶
  RastPort = PEEKL(WinBase+50)¶
  UserPort = PEEKL(WinBase+86)¶
RETURN¶
¶
CloseAll:¶
  CALL CloseWindow(WinBase)¶
  CALL UnDef¶
RETURN¶
¶
IntuitionMsg:¶
  MsgTyp    = PEEKL(IntuiMsg+20)¶
  Item      = PEEKL(IntuiMsg+28)¶
  GadgetNr% = PEEK(Item+39)¶
  CALL ReplyMsg(IntuiMsg)¶
¶
  IF (MsgTyp = GADGETDOWN) THEN¶
    'activated¶
    PRINT "DOWN Gadget-Nr.:";GadgetNr%¶
  END IF¶
¶
  IF (MsgTyp = GADGETUP) THEN¶
```

```
      'verify mpode¶
      PRINT "UP   Gadget-Nr.:";GadgetNr%;¶
      PRINT " Pos:";PEEKW(Buffer(GadgetNr%))¶
      Red = PEEKW(Buffer(1))¶
      Grn = PEEKW(Buffer(2))¶
      Blu = PEEKW(Buffer(3))¶
      PALETTE 1, Red/100, Grn/100, Blu/100¶
   END IF¶
¶
   IF (MsgTyp = CLOSEW) THEN¶
      'System-Gadget Window closer¶
      PRINT "CLOSE WINDOW"¶
      Info = 0¶
   END IF¶
RETURN¶
'-----------------------------------------¶
SUB DefChip(Buffer,Size)STATIC¶
 SHARED MList¶
  Size=Size+8¶
  Buffer=AllocMem(Size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,Size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
SUB UnDef STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList>0 THEN¶
    Address = PEEKL(MList)¶
    ListSize = PEEKL(MList+4)¶
    FreeMem MList, ListSize¶
    MList   = Address¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
SUB WinDef(bs, x%, y%, b%, h%, IDCMP, f, Gad, T$)
STATIC¶
  Size = 48+LEN(T$)+1¶
  DefChip bs,Size¶
  POKEW bs   ,x%       ' Angle gauche¶
  POKEW bs+ 2,y%       ' Angle supérieur¶
  POKEW bs+ 4,b%       ' Largeur¶
  POKEW bs+ 6,h%       ' Hauteur¶
  POKEW bs+ 8,65535&   ' Detail- BlockPen¶
  POKEL bs+10,IDCMP    ' Indicateurs IDCMP¶
  POKEL bs+14,f        ' Indicateurs¶
  POKEL bs+18,Gad      ' Premier Gadget¶
```

```
  POKEL bs+26,bs+48     ' Titre¶
  POKEW bs+46,1         ' Type d'écran¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+47+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB GadgetDef(bs, nx, x%, y%, b%, h%, f%, a%, T%, i,
Txt, si, n%) STATIC¶
  DefChip bs,44&        ' Longueur de Gadget-Structure¶
  POKEL bs   ,nx        '*Gadget suivant¶
  POKEW bs+ 4,x%        ' Angle gauche¶
  POKEW bs+ 6,y%        ' Angle supérieur¶
  POKEW bs+ 8,b%        ' Largeur¶
  POKEW bs+10,h%        ' Hauteur¶
  POKEW bs+12,f%        ' Indicateurs¶
  POKEW bs+14,a%        ' Activation¶
  POKEW bs+16,T%        ' Type du Gadget¶
  POKEL bs+18,i         ' Présentation du Gadget¶
  POKEL bs+26,Txt       '*Texte du Gadget¶
  POKEL bs+34,si        ' SpecialInfo¶
  POKEW bs+38,n%        ' GadgetID¶
END SUB¶
SUB IntuiText(bs, cl%, x%, y%, T$, nx) STATIC¶
  Size=20+LEN(T$)+1     ' Long. structure IntuiText +
Long. du Texte + Octet nul¶
  DefChip bs,Size¶
  POKE  bs   ,cl%       ' FrontPen¶
  POKE  bs+ 2,1         ' DrawMode¶
  POKEW bs+ 4,x%        ' Angle gauche¶
  POKEW bs+ 6,y%        ' Angle supérieur¶
  POKEL bs+12,bs+20     ' IText¶
  POKEL bs+16,nx        ' Texte suivant¶
  FOR i%=1 TO LEN(T$)¶
     POKE bs+19+i%,ASC(MID$(T$,i%,1))¶
  NEXT¶
END SUB¶
SUB Border(bs, x%, y%, c%, b%, h%) STATIC¶
  DefChip bs,48&        ' Long. de Border-Structure +
coordonnées¶
  POKEW bs  ,x%         ' Angle gauche¶
  POKEW bs+2,y%         ' Angle supérieur¶
  POKE  bs+4,c%         ' FrontPen¶
  POKE  bs+7,8          ' Compteur¶
  POKEL bs+8,bs+16      '*XY¶
  FOR i%=0 TO 1¶
     POKEW bs+22+i%*4,h%-1¶
     POKEW bs+24+i%*4,b%-1¶
     POKEW bs+32+i%*4,1¶
     POKEW bs+38+i%*4,h%-1¶
     POKEW bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
```

```
SUB PropInfo(bs, Flags%, HPot%, VPot%, HBody%,
VBody%) STATIC¶
  DefChip bs,22&¶
  POKEW bs   ,Flags%¶
  POKEW bs+ 2,HPot%¶
  POKEW bs+ 4,VPot%¶
  POKEW bs+ 6,HBody%¶
  POKEW bs+ 8,VBody%¶
END SUB¶
```

Intégration des nouveaux programmes

A l'issue de ce paragraphe, le développement de programmes conviviaux ne doit plus avoir de secret pour vous.

Nous avons tenté de mettre au point des procédures permettant d'accéder facilement au système d'exploitation et ,en particulier, à Intuition. Tous les programmes illustrès dans ce paragraphe sont écrits sous forme modulaire et peuvent donc être incorporés sans beaucoup d'efforts à vos propres programmes. Il faut procéder de la façon suivante :

Ecrivez chaque sous-programme dans un répertoire sous forme d'un fichier ASCII (save "nom du programme",A). Avant chaque routine, placez des commentaires énumérant les paramètres requis. Lorsque vous devez faire appel à Intuition, chargez le sous-programme à l'aide de Merge et placez l'appel dans le sous-programme OpenAll.

2.2. Le gestionnaire de fichiers

Après avoir exposé les principes de base de la programmation des accessoires graphiques et de l'accès à Intuition, nous aimerions vous montrer un autre programme qui tire lui aussi parti d'Intuition. Il est dix fois plus rapide d'accéder à l'écran et d'y afficher des données par l'intermédaire du système d'exploitation que sous BASIC.

Le gestionnaire de fichiers auquel ce paragraphe est consacré, vous permettra de visualiser, d'éditer et de modifier n'importe quel fichier disque sous forme ASCII ou hexadécimale. Le gestionnaire de fichiers utilise l'écran Intuition de façon à contrôler le programme par le biais de ses accessoires graphiques.

Listing du programme du gestionnaire de fichiers :

```
  REM DISKMON¶
  OPTION BASE 1¶
  DEFLNG a-z¶
  ON ERROR GOTO FAILED  :REM à enlever après test¶
  DECLARE FUNCTION ALLOCMEM LIBRARY¶
  DECLARE FUNCTION GETMSG LIBRARY¶
  LIBRARY"T&A2:bmaps/exec.library"¶
  LIBRARY"T&A2:bmaps/graphics.library"¶
  DECLARE FUNCTION OPENSCREEN LIBRARY¶
  DECLARE FUNCTION OPENWINDOW LIBRARY¶
  LIBRARY"T&A2:bmaps/intuition.library"¶
  DECLARE FUNCTION LOCK LIBRARY¶
  DECLARE FUNCTION EXAMINE LIBRARY¶
  DECLARE FUNCTION EXNEXT LIBRARY¶
  DECLARE FUNCTION IOERR LIBRARY¶
  DECLARE FUNCTION XOPEN LIBRARY¶
  DECLARE FUNCTION XREAD LIBRARY¶
  DECLARE FUNCTION XWRITE LIBRARY¶
  DECLARE FUNCTION SEEK LIBRARY¶
  LIBRARY"T&A2:bmaps/dos.library"¶
'  LPRINT :REM charge le driver d'imprimante au
démarrage¶
  PRINT"----------FILE MONITOR-V1.0----------"¶
  PRINT" '88 by DATA BECKER (w)'88 by S. M."¶
  PRINT¶
  PRINT"Le programme débute dans quelques secondes."¶
  PRINT"Patientez...(multitâche non autorisé"¶
  PRINT"pendant l'initialisation!)"¶
  DIM SHARED borders(14),itxt(25),gadgets(24),sinfo(2)¶
  bfec01=12577793&¶
  clearentry$=SPACE$(30)¶
  clearstring$=STRING$(80,0)¶
  INITIALIZE¶
  DIRECTORY¶
  start%=-1¶
  blocked%=0¶
  WHILE (-1)¶
    qualifier%=PEEK(bfec01)¶
    IF (qualifier%>&H60)AND(qualifier%<&H68)THEN¶
      IF qualifier% AND 1 THEN GOSUB keypressed¶
    END IF¶
    intuimsg=GETMSG(userport)¶
    IF intuimsg>0 THEN GOSUB IntuitionMsg¶
  WEND¶
IntuitionMsg:¶
  MsgTyp=PEEKL(intuimsg+20)¶
  IF MsgTyp=2097152& THEN¶
    IF start% THEN RETURN¶
    ascii.i%=PEEKW(intuimsg+24)¶
    IF ascii.i%>0 GOTO keypressed¶
```

```
  END IF¶
  Item=PEEKL(intuimsg+28)¶
  GadgetNr%=PEEK(Item+39)¶
  IF MsgTyp=32 THEN¶
    IF(GadgetNr%=10)OR(GadgetNr%=14)THEN blocked%=-1¶
    RETURN¶
  END IF¶
  IF MsgTyp<>64 THEN RETURN¶
  blocked%=0¶
  IF GadgetNr%<0 THEN ERROR 255¶
  IF GadgetNr%<5 THEN¶
    COPYMEM SADD(clearstring$),sinfo(1)+36,80¶
    POKEL sinfo(1)+36,CVL("DF"+CHR$(47+GadgetNr%)
+":")¶
    lasttype%=0¶
    DIRECTORY¶
  ELSEIF GadgetNr%<10 THEN¶
    SETFILEACTDIR¶
    IF lasttype%=1 THEN¶
      STATUS "Charge Bloc"¶
      OPENFILE¶
      IF oldhandle>0 THEN :STATUS "Edition"¶
      RETURN¶
    END IF¶
    DIRECTORY¶
  ELSEIF GadgetNr%=10 THEN¶
    blocked%=0¶
    DIRECTORY¶
    RETURN¶
  ELSEIF GadgetNr%=11 THEN¶
    IF dirstart%>4 THEN dirstart%=dirstart%-5:
DISPLAYDIR¶
  ELSEIF GadgetNr%=12 THEN¶
    IF number%>dirstart%+5 THEN¶
      dirstart%=dirstart%+5¶
      DISPLAYDIR¶
    END IF¶
  ELSEIF GadgetNr%=13 THEN¶
    DIRECTORY¶
  ELSEIF GadgetNr%=14 THEN¶
    blocked%=0¶
    newoffset=PEEKL(sinfo(2)+28)*488¶
    IF (newoffset>=newflen)OR(newoffset<0) THEN¶
      POKEL sinfo(2)+36,CVL("0"+MKI$(0)+CHR$(0))¶
      STATUS "Entrée interdite"¶
      DISPLAYBEEP scrbase¶
      RETURN¶
    END IF¶
    oldpos=SEEK(oldhandle,newoffset,-1)¶
    currentoffset=newoffset¶
    STATUS "Lecture du bloc"¶
    READBLOCK¶
```

```
    RETURN¶
  ELSEIF oldhandle=0 THEN¶
    POKEL sinfo(2)+36,CVL("0"+MKI$(0)+CHR$(0))¶
    STATUS "Aucun fichier"¶
    DISPLAYBEEP scrbase¶
    RETURN¶
  ELSEIF GadgetNr%=15 THEN¶
    COPYMEM fundo,fbuffer,488¶
    DISPLAYBUFFER¶
  ELSEIF GadgetNr%=16 THEN¶
    STATUS "Lecture"¶
    oldpos=SEEK(oldhandle,-amtread,0)¶
    READBLOCK¶
  ELSEIF GadgetNr%=17 THEN¶
    IF currentoffset<newflen-488 THEN¶
      STATUS "Secteur suivant"¶
      currentoffset=currentoffset+488¶
      READBLOCK¶
    END IF¶
  ELSEIF GadgetNr%=18 THEN¶
    IF currentoffset>487 THEN¶
      STATUS "Secteur précédent"¶
      currentoffset=currentoffset-488¶
      oldpos=SEEK(oldhandle,-amtread-488,0)¶
      READBLOCK¶
    END IF¶
  ELSEIF GadgetNr%=19 THEN¶
    STATUS "Ecrire en mémoire"¶
    oldpos=SEEK(oldhandle,-amtread,0)¶
    wr=XWRITE(oldhandle,fbuffer,amtread)¶
  ELSEIF GadgetNr%=20 THEN¶
    DUMPFILE¶
  ELSEIF GadgetNr%=21 THEN¶
    DUMPBUFFER¶
  ELSEIF GadgetNr%=22 THEN¶
    edmode%=0¶
     STATUS "Modifié en HEX"¶
  ELSEIF GadgetNr%=23 THEN¶
    edmode%=1¶
    STATUS "Modifié en ASCII"¶
  ELSEIF GadgetNr%=24 THEN¶
    STATUS "Vous partez ? O/N"¶
    t%=0¶
    WHILE (t%<>&HCE)AND(t%<>&H93)¶
      t%=PEEK(bfec01)¶
    WEND¶
    IF t%=&HCE THEN¶
      STATUS "Salut !"¶
      GOTO FAILED¶
    END IF¶
  END IF¶
  STATUS "OK"¶
```

```
  RETURN¶
keypressed:¶
  ascii$=UCASE$(CHR$(ascii.i%))¶
  IF edmode%=1 GOTO ASCIImode¶
  value%=INSTR("0123456789ABCDEF",ascii$)-1¶
  IF qualifier%=&H67 THEN¶
    offset%=offset%-24¶
    IF offset%<0 THEN offset%=amtread-1:nibble%=1¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H65 THEN¶
    offset%=offset%+24¶
    IF offset%>=amtread THEN offset%=0:nibble%=0¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H63 THEN¶
    IF nibble%=0 THEN¶
      nibble%=1¶
    ELSE¶
      nibble%=0¶
      offset%=offset%+1¶
      IF offset%>=amtread THEN offset%=0¶
    END IF¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H61 THEN¶
    IF nibble%=1 THEN¶
      nibble%=0¶
    ELSE¶
      nibble%=1¶
      offset%=offset%-1¶
      IF offset%<0 THEN offset%=amtread-1¶
    END IF¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  END IF¶
  IF value%>=0 THEN¶
    IF nibble%=0 THEN andi%=15:muls%=16:GOTO mk¶
    andi%=240¶
    muls%=1¶
mk: a%=(PEEK(fbuffer+offset%)AND andi%)+value%*muls%¶
    POKE fbuffer+offset%,a%¶
    CURSOROFF¶
    MOVE rastport,o.x%,o.y%+6¶
    SETAPEN rastport,1¶
    SETBPEN rastport,0¶
    TEXT rastport,SADD("0123456789ABCDEF")+value%,1¶
¶
```

```
    MOVE rastport,(o.b%+54)*8,o.y%+6¶
    SETAPEN rastport,0¶
    SETBPEN rastport,1¶
    TEXT rastport,fbuffer+offset%,1¶
¶
    IF nibble%=0 THEN nibble%=1:GOTO mk2¶
    nibble%=0¶
    offset%=offset%+1¶
    IF offset%>=amtread THEN offset%=0¶
mk2:¶
    CURSORON¶
    RETURN¶
  END IF¶
  RETURN¶
ASCIImode:¶
  IF qualifier%=&H67 THEN¶
    offset%=offset%-24¶
    IF offset%<0 THEN offset%=amtread-1:nibble%=1¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H65 THEN¶
    offset%=offset%+24¶
    IF offset%>=amtread THEN offset%=0:nibble%=0¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H63 THEN¶
    offset%=offset%+1:IF offset%>=amtread THEN
offset%=0¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  ELSEIF qualifier%=&H61 THEN¶
    offset%=offset%-1¶
    IF offset%<0 THEN offset%=amtread-1¶
    CURSOROFF¶
    CURSORON¶
    RETURN¶
  END IF¶
  IF ascii$<>CHR$(0) THEN¶
    value%=ascii.i%¶
    POKE fbuffer+offset%,value%¶
    CURSOROFF¶
¶
    MOVE rastport,o.x%+(o.m%=0)*nibble%,o.y%+6¶
    SETAPEN rastport,1¶
    SETBPEN rastport,0¶
    TEXT rastport,SADD(RIGHT$("0"+HEX$(value%),2)),2¶
¶
    SETAPEN rastport,0¶
    SETBPEN rastport,1¶
```

```
    MOVE rastport,(o.b%+54)*8,o.y%+6¶
    TEXT rastport,fbuffer+offset%,1¶
¶
¶
    offset%=offset%+1¶
    IF offset%>=amtread THEN offset%=0¶
    CURSORON¶
    RETURN¶
  END IF¶
  RETURN¶
¶
FAILED:¶
  UNDEF¶
  IF scrbase>0 THEN¶
    IF winbase>0 THEN¶
      CLOSEWINDOW winbase¶
      IF oldhandle>0 THEN :XCLOSE oldhandle¶
    END IF¶
    CLOSESCREEN scrbase¶
  END IF¶
  LIBRARY CLOSE¶
  END¶
'  SYSTEM¶
¶
SUB DUMPBUFFER STATIC¶
 SHARED fbuffer,HEXBUFF,currentlongs,currentoffset¶
  outstring$=SPACE$(1134)¶
  HEXBUFF currentlongs-1,fbuffer,SADD(outstring$)¶
  STATUS "J'imprime"¶
  FOR i%=0 TO 20¶
    LPRINT RIGHT$("      "+STR$(currentoffset+i%*20),
8);": ";¶
    LPRINT MID$(outstring$,i%*54+1,54)¶
 NEXT¶
 LPRINT¶
END SUB¶
SUB DUMPFILE STATIC¶
 SHARED amtread,oldhandle,currentoffset¶
  savedoffset=currentoffset¶
  oldpos=SEEK(oldhandle,0,-1)¶
  currentoffset=0¶
df.loop:¶
  READBLOCK¶
  DUMPBUFFER¶
  currentoffset=currentoffset+488¶
  IF amtread=488 GOTO df.loop¶
  currentoffset=savedoffset¶
  oldpos=SEEK(oldhandle,currentoffset,-1)¶
  READBLOCK¶
END SUB¶
¶
SUB CURSORON STATIC¶
```

```
 SHARED o.x%,o.y%,edmode%,o.m%,rastport,offset%,
nibble%¶
 SHARED o.b%¶
  z%=INT(offset%/24)¶
  o.b%=offset%-z%*24¶
  l%=INT(o.b%/4)¶
  o.x%=(o.b%*2+l%-(edmode%=0)*nibble%)*8¶
  o.y%=z%*8+2¶
  SETAPEN rastport,3¶
  SETDRMD rastport,3¶
¶
  RECTFILL rastport,o.x%,o.y%,o.x%+7-(edmode%=1)*8,
o.y%+7¶
  RECTFILL rastport,(o.b%+54)*8,o.y%,(o.b%+54)*8+7,
o.y%+7¶
  SETDRMD rastport,1¶
  o.m%=edmode%¶
END SUB¶
¶
SUB CURSOROFF STATIC¶
 SHARED o.x%,o.y%,o.m%,o.b%,rastport¶
  SETAPEN rastport,3¶
  SETDRMD rastport,3¶
¶
  RECTFILL rastport,o.x%,o.y%,o.x%+7-(o.m%=1)*8,
o.y%+7¶
  RECTFILL rastport,(o.b%+54)*8,o.y%,(o.b%+54)*8+7,
o.y%+7¶
¶
  SETDRMD rastport,1¶
END SUB¶
¶
SUB OPENFILE STATIC¶
 SHARED oldhandle,scrbase,currentoffset,actdir,
newflen¶
 SHARED numblocks¶
  IF oldhandle>0 THEN :XCLOSE oldhandle¶
  oldhandle=XOPEN(actdir,1005)¶
  IF oldhandle=0 THEN¶
    STATUS "Open a echoué"¶
    DISPLAYBEEP scrbase¶
    EXIT SUB¶
  END IF¶
  numblocks=newflen/488¶
  w=CVL(RIGHT$("  "+STR$(numblocks),4))¶
  POKEL itxt(12)+20,w¶
  currentoffset=0¶
  READBLOCK¶
END SUB¶
SUB READBLOCK STATIC¶
 SHARED oldhandle,fbuffer,fundo,amtread,currentlongs¶
 SHARED currentoffset¶
```

```
  amtread=XREAD(oldhandle,fbuffer,488)¶
  IF amtread<488 THEN¶
    v$=STRING$(488-amtread,0)¶
    COPYMEM SADD(v$),fbuffer+amtread,LEN(v$)¶
  END IF¶
  x=currentoffset/488¶
  w=CVL(LEFT$(MID$(STR$(x),2)+MKL$(0),4))¶
  POKEL sinfo(2)+36,w¶
¶
  currentlongs=(amtread+3)/4¶
  COPYMEM fbuffer,fundo,488¶
  DISPLAYBUFFER¶
END SUB¶
SUB DISPLAYBUFFER STATIC¶
 SHARED
HEXBUFF,currentlongs,fbuffer,rastport,amtread¶
 SHARED start%,offset%,nibble%¶
  ASCIIbuffer$=SPACE$(1134)¶
  HEXBUFF currentlongs-1,fbuffer,SADD(ASCIIbuffer$)¶
  SETAPEN rastport,0¶
  RECTFILL rastport,0,0,639,140¶
  SETAPEN rastport,1¶
  SETBPEN rastport,0¶
  FOR i%=0 TO 20¶
    MOVE rastport,0,i%*8+8¶
    TEXT rastport,SADD(ASCIIbuffer$)+i%*54,54¶
  NEXT¶
  SETAPEN rastport,0¶
  SETBPEN rastport,1¶
  l%=24¶
  FOR i%=0 TO 20¶
    MOVE rastport,432,i%*8+8¶
    IF i%=20 THEN l%=8¶
    TEXT rastport,fbuffer+i%*24,l%¶
  NEXT¶
  start%=0¶
  offset%=0¶
  nibble%=0¶
  CURSORON¶
END SUB¶
SUB SETFILEACTDIR STATIC¶
 SHARED GadgetNr%,actdir,scrbase,clearstring$,newflen¶
 SHARED dirstart%,dirbuff,lasttype%¶
  compare1$=STRING$(31,0)¶
  compare2$=STRING$(80,0)¶
  COPYMEM actdir,SADD(compare2$),79¶
  COPYMEM itxt(GadgetNr%)+20,SADD(compare1$),30¶
  l2%=INSTR(compare2$,CHR$(0))-1¶
  l1%=INSTR(compare1$,CHR$(0))-1¶
  IF lasttype%=1 THEN¶
    l2%=INSTR(compare2$,":")¶
path.loop:¶
    l3%=INSTR(l2%+1,compare2$,"/")¶
```

```
    IF l3%>l2% THEN l2%=l3%:GOTO path.loop¶
  END IF¶
  IF(l1%+l2%)>78 THEN¶
    STATUS "Nom trop long"¶
    DISPLAYBEEP scrbase¶
    EXIT SUB¶
  END IF¶
  v$=LEFT$(compare2$,l2%)¶
  IF(lasttype%>1)AND(RIGHT$(v$,1)<>":")THEN v$=v$+"/"¶
  lasttype%=PEEK(dirbuff+(dirstart%+GadgetNr%-5)*
36+31)¶
  v$=LEFT$(v$+compare1$+clearstring$,79)¶
  COPYMEM SADD(v$),actdir,79¶
  newflen=PEEKL(dirbuff+(dirstart%+GadgetNr%-5)*
36+32)¶
END SUB¶
SUB DISPLAYDIR STATIC¶
 SHARED dirstart%,number%,clearentry$,dirbuff,winbase¶
  FOR i%=5 TO 9¶
    COPYMEM SADD(clearentry$),itxt(i%)+20,30¶
  NEXT¶
  i%=0¶
  IF number%<=dirstart% GOTO displaydir.show¶
  REFRESHGADGETS gadgets(23),winbase,0¶
displaydir.loop:¶
  a=dirbuff+(i%+dirstart%)*36¶
  COPYMEM a,itxt(i%+5)+20,30¶
  POKE itxt(i%+5),PEEK(a+31)¶
  i%=i%+1¶
  IF (i%<5)AND(number%>(dirstart%+i%))GOTO displaydir.
loop¶
displaydir.show:¶
  REFRESHGADGETS gadgets(23),winbase,0¶
END SUB¶
SUB DIRECTORY STATIC¶
 SHARED number%,dirstart%,actdir,lasttype%¶
 SHARED fileinfo,clearentry$,dirbuff,newflen¶
  STATUS "J'examine l'entrée"¶
  dirlock=LOCK(actdir,-2)¶
  IF dirlock=0 THEN¶
    STATUS "Fichier non trouvé"¶
    EXIT SUB¶
  END IF¶
  e=EXAMINE(dirlock,fileinfo)¶
  IF e=0 THEN¶
    UNLOCK dirlock¶
    STATUS "Erreur à l'examen"¶
    EXIT SUB¶
  END IF¶
  IF PEEKL(fileinfo+120)<0 THEN¶
    newflen=PEEKL(fileinfo+124)¶
    UNLOCK dirlock¶
```

```
    OPENFILE¶
    lasttype%=1¶
    EXIT SUB¶
  END IF¶
  lasttype%=3¶
  number%=0¶
  dirstart%=0¶
  FOR i%=5 TO 9¶
    COPYMEM SADD(clearentry$),itxt(i%)+20,30¶
  NEXT¶
  STATUS "Lect. Répertoire"¶
directory.loop:¶
  e=EXNEXT(dirlock,fileinfo)¶
  IF e=0 THEN¶
    e=IOERR¶
    IF e<>232 THEN¶
      STATUS "Répertoire faux"¶
      number%=0¶
    ELSE¶
      STATUS "Ok"¶
    END IF¶
    UNLOCK dirlock¶
    DISPLAYDIR¶
    EXIT SUB¶
  END IF¶
  a=dirbuff+number%*36¶
  COPYMEM fileinfo+8,a,30¶
  IF PEEKL(fileinfo+120)<0 THEN c%=1 ELSE c%=3¶
  POKE a+31,c%¶
  POKEL a+32,PEEKL(fileinfo+124)¶
  number%=number%+1¶
  IF number%<72 GOTO directory.loop¶
  UNLOCK dirlock¶
  STATUS "Okay"¶
  DISPLAYDIR¶
END SUB¶
SUB INITIALIZE STATIC¶
 SHARED HEXBUFF,fbuffer,fundo,nscreen,dirbuff,
fileinfo¶
 SHARED actdir,scrbase,winbase,viewport,rastport¶
 SHARED userport¶
  FORBID¶
  DEFCHIP HEXBUFF,60&¶
  DEFCHIP fbuffer,488&¶
  DEFCHIP fundo,488&¶
  DEFCHIP nscreen,88&¶
  DEFCHIP dirbuff,2592&¶
  DEFCHIP fileinfo,252&¶
  DEFCHIP shows,68&¶
  borders(13)=shows+28¶
  borders(14)=shows+48¶
  FOR i%=0 TO 14¶
```

```
    READ i$¶
    POKEW HEXBUFF+i%*4,VAL("&H"+LEFT$(i$,4))¶
    POKEW HEXBUFF+i%*4+2,VAL("&H"+RIGHT$(i$,4))¶
  NEXT¶
  FOR i%=0 TO 6¶
    READ i$¶
    POKEW shows+i%*4,VAL("&H"+LEFT$(i$,4))¶
    POKEW shows+i%*4+2,VAL("&H"+RIGHT$(i$,4))¶
  NEXT¶
  POKE shows+29,10¶
  POKE shows+31,3¶
  POKE shows+33,7¶
  POKE shows+35,7¶
  POKE shows+37,1¶
  POKEW shows+42,256¶
  COPYMEM shows+28,shows+48,20¶
  FOR i%=0 TO 1¶
    POKEL shows+i%*20+38,shows+i%*14¶
  NEXT¶
  FOR i%=1 TO 12¶
    READ a%,b%,c%,d%,e%,f%¶
    BORDER borders(i%),a%,b%,c%,d%,e%¶
    IF f%>0 THEN POKEL borders(i%)+12,borders(f%)¶
  NEXT¶
  FOR i%=1 TO 4¶
    IntuiText itxt(i%),1,6,3,"DF"+CHR$(47+i%)+":",0&¶
  NEXT¶
  FOR i%=5 TO 9¶
    IntuiText itxt(i%),1,8,0,SPACE$(30),0&¶
  NEXT¶
  FOR i%=10 TO 25¶
    READ a%,b%,c%,d$,e%¶
    IF e%>0 THEN f=itxt(e%) ELSE f=0¶
    IntuiText itxt(i%),a%,b%,c%,d$,f¶
  NEXT¶
  STRINGINFO sinfo(1),79,"DF0:"¶
  STRINGINFO sinfo(2),4,"0"+STRING$(15,0)¶
  actdir=sinfo(1)+36¶
  d=0¶
  FOR i%=1 TO 24¶
    READ e%,f%,G%,h%,j%,k%,l%,m%,n%,o%¶
    IF o%>0 THEN a=sinfo(o%) ELSE a=0¶
    IF n%>0 THEN b=itxt(n%) ELSE b=0¶
    IF m%>0 THEN c=borders(m%) ELSE c=0¶
    GADGET gadgets(i%),d,e%,f%,G%,h%,j%,k%,l%,c,b,a,
i%¶
    d=gadgets(i%)¶
  NEXT i%¶
  POKEL nscreen+4,41943296&¶
  POKE nscreen+9,2¶
  POKE nscreen+12,192¶
  POKEW nscreen+14,&H10F¶
  nwindow=nscreen+32¶
```

```
  POKEL nwindow+4,41943296&¶
  POKEW nwindow+8,259¶
  POKE nwindow+11,32¶
  POKE nwindow+13,96¶
  POKE nwindow+15,1¶
  POKE nwindow+16,24¶
  POKEL nwindow+18,d¶
  POKE nwindow+47,15¶
  POKEW nscreen+82,&HFFF¶
  POKE nscreen+84,15¶
  POKEW nscreen+86,&HFD0¶
  PERMIT¶
  scrbase=OPENSCREEN(nscreen)¶
  IF scrbase=0 THEN ERROR 7¶
  POKEL nwindow+30,scrbase¶
  winbase=OPENWINDOW(nwindow)¶
  IF winbase=0 THEN ERROR 7¶
  rastport=PEEKL(winbase+50)¶
  viewport=scrbase+44¶
  userport=PEEKL(winbase+86)¶
  LOADRGB4 viewport,nscreen+80,4¶
END SUB¶
SUB STATUS(t$)STATIC¶
 SHARED winbase¶
  t$=LEFT$(t$+SPACE$(17),17)¶
  COPYMEM SADD(t$),itxt(22)+20,17¶
  REFRESHGADGETS gadgets(23),winbase,0¶
END SUB¶
SUB DEFCHIP(Buffer,size)STATIC¶
 SHARED MList¶
  size=size+8¶
  Buffer=ALLOCMEM(size,65538&)¶
  IF Buffer>0 THEN¶
    POKEL Buffer,MList¶
    POKEL Buffer+4,size¶
    MList=Buffer¶
    Buffer=Buffer+8¶
  ELSE¶
    ERROR 7¶
  END IF¶
END SUB¶
SUB UNDEF STATIC¶
 SHARED MList¶
undef.loop:¶
  IF MList>0 THEN¶
    Buffer=PEEKL(MList)¶
    size=PEEKL(MList+4)¶
    FREEMEM MList,size¶
    MList=Buffer¶
    GOTO undef.loop¶
  END IF¶
END SUB¶
```

```
SUB GADGET(bs,nx,x%,y%,b%,h%,f%,a%,t%,i,txt,si,n%)
STATIC¶
  DEFCHIP bs,44&¶
  POKEL bs,nx¶
  POKEW bs+4,x%¶
  POKEW bs+6,y%¶
  POKEW bs+8,b%¶
  POKEW bs+10,h%¶
  POKEW bs+12,f%¶
  POKEW bs+14,a%¶
  POKEW bs+16,t%¶
  POKEL bs+18,i¶
  POKEL bs+26,txt¶
  POKEL bs+34,si¶
  POKEW bs+38,n%¶
END SUB¶
SUB IntuiText(bs,c1%,x%,y%,t$,nx)STATIC¶
  size=20+LEN(t$)+1¶
  DEFCHIP bs,size¶
  POKE bs,c1%¶
  POKE bs+2,1¶
  POKEW bs+4,x%¶
  POKEW bs+6,y%¶
  POKEL bs+12,bs+20¶
  POKEL bs+16,nx¶
  COPYMEM SADD(t$),bs+20,LEN(t$)¶
END SUB¶
SUB BORDER(bs,x%,y%,c%,b%,h%)STATIC¶
  DEFCHIP bs,48&¶
  POKEW bs,x%¶
  POKEW bs+2,y%¶
  POKE bs+4,c%¶
  POKE bs+7,8¶
  POKEL bs+8,bs+16¶
  FOR i%=0 TO 1¶
    POKEW bs+22+i%*4,h%-1¶
    POKEW bs+24+i%*4,b%-1¶
    POKEW bs+32+i%*4,1¶
    POKEW bs+38+i%*4,h%-1¶
    POKEW bs+40+i%*4,b%-2¶
  NEXT¶
END SUB¶
SUB STRINGINFO(bs,max%,buff$)STATIC¶
  IF LEN(buff$)>max% THEN nmax%=LEN(buff$) ELSE
nmax%=max%¶
  IF(nmax%AND 1)THEN nmax%=nmax%+1¶
  size=36+2*(nmax%+4)¶
  DEFCHIP bs,size¶
  POKEL bs,bs+36¶
  POKEL bs+4,bs+40+nmax%¶
  POKEW bs+10,max%+1¶
  IF buff$<>""THEN¶
```

```
    COPYMEM SADD(buff$),bs+36,LEN(buff$)¶
  END IF¶
END SUB¶
DATA 48E7F0C0,4CEF0308,001C5303,22187407,E9991001,0200000F¶
DATA 06000030,0C00003A,65040600,000712C0,51CAFFE6,12FC0020¶
DATA 51CBFFDA,4CDF030F,4E750000,10003800,7C00FE00,38003800¶
DATA 38003800,38003800,FE007C00,38001000¶
'¶
DATA 0,0,2,43,13,0,-6,-3,2,268,45,0,-6,-3,3,268,13,0¶
DATA 0,0,2,28,13,0,0,-45,2,28,13,4,0,-15,2,28,13,5¶
DATA -62,-3,2,172,13,0,0,0,2,65,13,0,0,0,2,109,13,0¶
DATA 0,15,2,218,13,9,0,0,2,60,13,0,0,0,2,43,28,0¶
DATA 3,-56,0,"Bloc:",0,3,40,0,"de:",10,1,72,0,"   0",11¶
DATA 3,6,3,"OK",0,1,6,3,"UNDO",0,1,6,3,"Imprime Bloc",0¶
DATA 1,6,3,"Imprime Fich",0,1,17,3,"LIRE",0,1,17,3,"AVANT",0¶
DATA 1,13,3,"RETOUR",0,1,13,3,"ECRIRE",0,3,6,18,"Statut:",15¶
DATA 1,68,18,"Lect. Répertoire",21,1,9,3,"ASCII",0¶
DATA 1,9,3," HEX",0,1,6,10,"Fin",0¶
'¶
DATA 0,198,43,13,0,3,1,1,1,0,0,213,43,13,0,3,1,1,2,0¶
DATA 0,228,43,13,0,3,1,1,3,0,0,243,43,13,0,3,1,1,4,0¶
DATA 52,201,256,8,0,3,1,2,5,0,52,209,256,8,0,3,1,0,6,0¶
DATA 52,217,256,8,0,3,1,0,7,0,52,225,256,8,0,3,1,0,8,0¶
DATA 52,233,256,8,0,3,1,0,9,0,52,246,256,8,0,3,4,3,0,1¶
DATA 317,198,28,13,4,3,1,13,0,0,317,228,28,13,4,3,1,14,0,0¶
DATA 317,243,28,13,0,3,1,6,13,0¶
DATA 416,201,40,8,0,2051,4,7,12,2¶
DATA 529,198,43,13,1,3,1,1,14,0¶
DATA 575,198,65,13,0,3,1,8,17,0,575,213,65,13,0,3,1,8,18,0¶
DATA 575,228,65,13,0,3,1,8,19,0,575,243,65,13,0,3,1,8,20,0¶
DATA 354,213,109,13,0,3,1,9,16,0¶
DATA 354,228,109,13,0,3,1,10,22,0¶
DATA 466,213,60,13,0,3,1,11,24,0¶
DATA 466,228,60,13,0,3,1,11,23,0¶
DATA 529,213,43,28,128,3,1,12,25,0¶
DATA 0,0¶
```

2.2.1. Utilisation du gestionnaire de fichiers

Ce gestionnaire de fichier utilise une grande quantité de mémoire vive (RAM). Cela signifie que vous ne pouvez exécuter qu'une seule tâche à la fois lorsque vous utilisez les fonctions du gestionnaire. En d'autres termes, il n'est pas possible de faire tourner d'autres programmes en parallèle. Si vous lancez un second programme simultanément, la mémoire vive (RAM) risque de vous faire défaut. La position de LPRINT en début de programme assure le chargement du driver d'imprimante en mémoire avant l'affectation d'un espace mémoire au programme. Si vous n'utilisez pas d'imprimante, cette ligne est inutile.

Les quatre Gadgets de message, qui représentent les quatre lecteurs les plus fréquemment utilisés, peuvent être modifiés en éditant les instructions DATA correspondantes. Ces Gadgets sont utilisés par la routine de répertoire. Pour sélectionner un lecteur, il suffit de cliquer sur le Gadget approprié.

Les lecteurs par défaut du programme vont de DF0: à DF3:. Si vous désirez voir apparaître d'autres lecteurs à l'intérieur des Gadgets, modifiez les noms en conséquence dans les instructions DATA en prenant soin de leur attribuer un nom de quatres caractères maximum (les deux points finals compris). Vous pourrez leur attribuer les noms des lecteurs DF0: à DF3, à l'aide de Assign, avant de charger le gestionnaire.

Les quatre Gadgets situés à gauche de l'écran permettent d'accélerer la sélection du lecteur et des répertoires. Pour visualiser le répertoire principal du lecteur interne, cliquez sur le Gadget DF0:.

La liste affiche jusqu'à cinq enregistrements de répertoires. Les fichiers et les programmes apparaissent en blanc et les répertoires en jaune. Si l'on clique sur le nom d'un répertoire, son contenu s'affiche. Si l'on clique sur un fichier, le premier bloc de données du fichier se charge en mémoire puis apparaît à l'écran. Ce bloc de données peut être édité soit sous forme ASCII, soit sous forme héxadécimale. La zone de texte qui figure sous la liste des fichiers indique le répertoire ou le nom de fichier courant. Si vous cliquez dessus, un curseur apparaît. Vous pouvez alors entrer les noms de fichier/chemins qui vous intéressent à partir du clavier. Cette zone est particulièrement utile lorsque l'on désire entrer un long chemin d'accès ou que l'on veut accéder à un lecteur qui n'est pas représenté sur l'écran.

La flèche, à droite de la liste de fichiers, vous permet de faire défiler la totalité de la liste et de visualiser ainsi l'ensemble des fichiers. Les fichiers sont décalés par groupe de cinq.

Cliquer sur OK valide le nom de fichier inscrit dans la zone de texte. Cela revient en fait à appuyer sur la touche <Return>, une fois la modification de la zone de texte est achevée.

La ligne Bloc #### de: #### indique le numéro du bloc affiché à l'écran ainsi que le nombre total de blocs du fichier sélectionné. En cliquant sur le premier nombre, on peut le modifier de façon à obtenir le numéro du bloc désiré.

Les deux cases Imprime permettent d'imprimer soit le tampon d'édition soit le fichier entier sous forme héxadécimale. Lorsque l'impression est terminée, le dernier bloc édité réapparaît à l'écran. Status affiche l'ensemble des erreurs et les opérations courantes. Si tout fonctionne correctement, le message OK apparaît.

Les Gadgets ASCII et HEX permettent de sélectionner un affichage sous forme ASCII ou héxadécimale. Cette option est particulièrement intéressante pour modifier un texte (par exemple, pour personnaliser des menus AmigaBASIC). La case Quit et le message de confirmation qui l'accompagne mettent fin au programme.

Les Gadgets LIRE, AVANT et SUITE permettent de lire respectivement le bloc courant, le bloc suivant et le bloc précédent. Le Gadget Write écrit le tampon d'édition sur le disque. Aucun message de confirmation n'apparaît (afin d'accélérer l'opération). Si vous écrivez un bloc par erreur, sélectionnez le Gadget Undo puis à nouveau le Gadget Write.

La mémoire tampon de Undo conserve le contenu original des données en cours d'affichage. Le Gadget Undo place le contenu de ce tampon dans le tampon d'édition.

L'éditeur accepte tous les caractères qui peuvent être entrès à partir du clavier , y compris les touches de déplacement du curseur.

Le programme peut tourner en parallèle avec d'autres applications ,bien qu'en réalité cela ne soit guère conseillé (voir nos recommendations au début du chapitre). La combinaison de la touche <Amiga> de gauche avec les touches , et N (<Amiga><,> et <Amiga><N>) permettent de basculer du gestionnaire de fichier à l'écran Workbench.

Un dernier point : le gestionnaire de fichier ne peut lire un chemin au delà de deux niveaux de répertoire. Si vous désirez améliorer votre marge de manoeuvre, il vous faudra redéfinir la taille de la mémoire tampon du répertoire en conséquence et rectifier le sous-programme du répertoire. Vous pouvez accéder directement à chaque fichier au moyen de la zone de texte.

2.2.2. Modifier des fichiers à l'aide du moniteur

Il est possible de modifier un programme existant en manipulant certains de ses octets. Cette opération a pour objectif d'adapter les programmes aux besoins de l'utilisateur.

Il ne faut cependant pas perdre de vue que la loi interdit la modification des messages de copyright ou la copie de programmes commercialisés. Pour rester dans les limites de la légalité, ne modifiez des programmes distribués sur le marché que pour votre usage personnel. D'autre part, ne modifiez en aucun cas les messages de copyright et n'utilisez pas ce gestionnaire de fichier à des fins proscrites par la loi.

2.2.3. Modification de AmigaBASIC

Vous pouvez personnaliser votre copie d'AmigaBASIC, lui donner une touche personnelle, en modifiant les menus et les messages d'erreur à l'aide du gestionnaire de fichier.

Attention : Toutes les fois que vous voudrez modifier un programme ou éditer un fichier, travaillez sur des copies, jamais sur des originaux !

Commencez par copier AmigaBASIC sur une autre disquette. Exécutez le programme du gestionnaire de fichier et sélectionnez le mode ASCII. Insérez ensuite la disquette contenant la copie d'AmigaBASIC (et NON PAS l'original!). Sélectionnez AmigaBASIC et appuyez sur la touche <Return>.

Dès que le fichier est chargé, cliquez sur le Gadget Block, entrez 28 et appuyez sur <Return>. Utilisez ensuite les Gadgets Next et Back pour feuilleter AmigaBASIC jusquà ce que vous trouviez les menus. Utilisez les touches fléchées pour déplacer le curseur et modifier le fichier de manière à personnaliser AmigaBASIC. Cliquez sur Write pour sauvegarder les modifications apportées. En cas de fausse manoeuvre, cliquez sur Undo, rectifier l'erreur et cliquez à nouveau sur Write. Enfin, cliquez sur Quit et appuyez sur <O> pour quitter le programme.

Quelques indications sur la façon de modifier vos menus sont sans doute les bienvenues. Voici des exemples de ce que nous avons fait pour changer nos menus AmigaBASIC à l'aide du programme de gestion de fichier.

Menus originaux			
Project	Edit	Run	Windows
New	Cut	Start	Show List
Open	Copy	Stop	Show Output
Save	Paste	Continue	
Save as		Suspend	
Quit		Trace on	
		Step	

Menus modifiés			
Stuff	Edit	Run	Screens
Oops	Cut	Go	List
Load it	Copy	Break	BASIC
Save it	Paste	Keep on	
Save as		Whoa	
Ciao		Trace on	
		Step	

2.3. Accéder à AmigaDOS à partir d'AmigaBASIC

Le système d'exploitation d'Amiga consiste en plusieurs petites fonctions dont l'action conjuguée permet d'accomplir des tâches plus importantes. L'une des activités essentielles est l'accès au disque. Lors de toute opération d'accès au disque, Amiga sollicite les routines de gestion de disque à l'intérieur du système d'exploitation. AmigaBASIC n'échappe pas à cette règle.

Peu d'utilisateurs savent combien ces routines sont simples à utiliser sous AmigaBASIC. La première étape consiste à ouvrir le fichier dos.library pour avoir accès au DOS (Disk Operating System). Pour exploiter les programmes contenus dans ce fichier, il vous faut convertir le fichier

dos_lib.fd, enregistré dans le répertoire FD de votre disquette Extras, en un fichier .bmap. Le programme ConvertFD (dans le répertoire BasicDemos de la diquette Extras) convertit les fichiers _lib.fd en des fichiers dos.bmap. Le système d'exploitation peut lire le fichier dos_lib.fd, mais AmigaBASIC ne peut pas lire les données sous cette forme. Le fichier .bmap transforme donc les données DOS en un format compréhensible par AmigaBASIC.

Remarque :

Nos fichiers .bmap proviennent de la version 1.3 du système d'exploitation. Vous pouvez eventuellement utiliser la version 1.2. Nous vous conseillons cependant d'acquérir Workbench 1.3 et Extras 1.3 si vous ne les possédez pas encore.

La ligne de commande permettant d'accéder à ce fichier à partir d'AmigaBASIC est la suivante :

```
LIBRARY "dos.library"¶
```

Remarquez le nom du fichier. Le fichier stocké sur le disque doit être dos.bmap ; le nom de fichier utilisé sous AmigaBASIC doit être dos.library

Avant de commencer à programmer, examinons plus en détail le contenu du fichier library.dos. Voici une liste des commandes les plus importantes.

Accès aux fichiers

```
xOpen (nom, mode d'accès)
Close (fichier)
xRead (fichier)
Write (fichier, mémoire tampon, longueur)
Seek (fichier, mémoire tampon, longueur)
Lock (fichier, position, décalage)
Unlock (verrou)
DupLock (verrou)
```

Gestion des fichiers

```
DeleteFile (nom)
Rename(ancien nom, nouveau nom)
Examine( verrou, bloc info. du fichier)
```

```
ExNext(verrou, bloc info. du fichier)
Info(verrou, bloc de paramètre)
CreateDir(nom)
CurrentDir(verrou)
SetComment(nom, commentaire)
SetProtection(nom, masque)
DateStamp(date)
ParentDir(verrou)
```

Gestion du processus

```
Exit (code de retour)
LoadSeg(nom du fichier)
UnLoadSeg(segment)
IsInteractive(fichier)
Execute(chaîne, fichier, fichier)
DeviceProc(nom)
Delay(délai)
```

Cette liste est loin d'être exhaustive. Les commandes répertoriées sont celles qui présentent le plus d'interêt pour les programmeurs en BASIC. Les fonctions de gestion du processus ne sont pas réellement d'un accès facile à partir de BASIC. C'est pourquoi, nous accorderons plus d'attention aux deux premiers groupes de commandes.

Les commandes du premier groupe permettent d'accéder aux fichiers séquentiels ou relatifs sous AmigaBASIC. Les fonctions sont plus nombreuses que celles utilisées sous BASIC. Nous utiliserons quelques unes d'entre elles très bientôt.

Les commandes du second groupe permettent de gérer la structure de disque. Cette structure comprend les répertoires, les fichiers et les modifications qui leur sont apportées.

2.3.1. Accès aux fichiers sous AmigaDOS

Examinez attentivement les commandes d'accès aux fichiers de dos.library. Vous y trouverez les fonctions Open() et Close(). Ces fonctions utilisent des arguments totalement différents. Analysons les plus en détail.

Le programme suivant utilise les trois premières fonctions (et aussi les plus importantes) AmigaDOS de dos.library. Il charge les 200 premiers octets d'un fichier ASCII quelconque dans une mémoire tampon et affiche les données sous forme ASCII. Entrez le programme et enregistrez le sous format ASCII comme suit :

```
SAVE "xopendemo",a¶
```

Exécutez le programme. Il lira son contenu et affichera ses 200 premiers octets sur l'écran. Voici le listing :

```
'*********************************¶
'*   Demo de xOpen               *¶
'*   Test de la fonction xOpen   *¶
'* --------------------------- *¶
'*                               *¶
'* Auteur : Wolf-Gideon Bleek    *¶
'* Date   : July 9, 1988         *¶
'* Version: 1.0                  *¶
'* System d'expl. : V1.2 & V1.3  *¶
'*                               *¶
'*********************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION xOpen& LIBRARY¶
DECLARE FUNCTION xRead& LIBRARY¶
'                xClose¶
¶
DIM Memory%(100)¶
¶
WIDTH 70¶
¶
File$ = "xOpendemo"+CHR$(0)¶
¶
Handle& = xOpen&(SADD(File$), 1005)¶
¶
Amount& = xRead&(Handle&, VARPTR(Memory%(0)), 200&)¶
¶
CALL xClose(Handle&)¶
¶
¶
FOR i = 0 TO 200¶
   PRINT CHR$(PEEK(VARPTR(Memory%(0))+i));¶
NEXT i¶
¶
LIBRARY CLOSE¶
```

Description des commandes

La fonction xOpen() correspond à l'instruction OPEN d'AmigaBASIC. Les fonctions portent un nom différent pour éviter que le programmeur ne les confonde. xOpen() necéssite un pointeur sur une chaîne constituée d'un nom de fichier et d'un mode d'accès. Les élements nécessaires pour accéder au fichier sont ainsi réunis. Si une erreur se produit, la fonction renvoie une valeur nulle.

A l'aide de ces éléments, nous pouvons accéder au fichier ouvert avec la fonction xRead(). Encore une fois, cette fonction porte un nom différent dans le fichier .bmap afin de ne pas la confondre avec la fonction READ en BASIC. xRead() attend les arguments communiqués par la fonction xOpen() pour savoir à quel fichier accéder ,plusieurs fichiers pouvant être ouverts simultanément. Ceci est similaire aux numéros logiques de fichiers utilisés sous AmigaBASIC.

Nous devons transférer l'adresse d'une mémoire tampon à partir de laquelle les données peuvent être lues. Nous avons construit un tableau pouvant contenir jusqu'à 200 octets. Ce tableau accepte 100 variables entières de deux octets chacune. Le dernier argument représente le nombre d'octets devant être lus. Cette valeur ne doit pas dépasser la taille de la mémoire tampon sous peine de voir les données placées dans un endroit de la mémoire utilisée à d'autres tâches et de provoquer du même coup un incident système.

Quand la fonction est exécutée, elle place les 200 premiers octets du fichier ouvert dans la mémoire tampon. La fonction xClose() ferme le fichier et le programme se termine par l'affichage de ces 200 octets sur l'écran.

Vous vous demandez peut être encore en quoi ces trois fonctions de fichier peuvent vous être utiles. Un petit exemple vous éclairera. Un programme gère plusieurs algorithmes à la fois. Un tableau de 100 x 100 éléments est utilisé pour ranger ces algorithmes qui y sont conservés pour une utilisation ultèrieure, le temps de calcul pouvant durer cinq heures. Par conséquent,il vous faut mettre au point une routine de stockage de ces algorithmes sur le disque. Vous pouvez soit écrire les nombres un par un à l'aide de PRINT#, soit enregsitrer l'adresse en mémoire du tableau pour réaliser une sauvegarde sur disque à l'aide de la fonction xWrite(). Cette dernière méthode vous permet de recharger l'ensemble des valeurs à l'aide de la commande xRead.

Le programme ,ci-dessous, illustre la procédure qui vient d'être décrite en prenant comme base un tableau constitué de 10 x 10 éléments:

```
'*****************************************¶
'*                                       *¶
'* Demo de xWrite                        *¶
'* Sauvegarde des données avec xWrite()  *¶
'* ------------------------------------- *¶
'*                                       *¶
'* Auteur : Wolf-Gideon Bleek            *¶
'* Date   : July 10,1988                 *¶
'* Version: 1.0                          *¶
'* Système d'expl. : V1.2 & V1.3         *¶
'*                                       *¶
'*****************************************¶
¶
OPTION BASE 1¶
'les bibliothèques sont sur la disquette T&A2, les
vôtres se trouvent peut-être ailleurs¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION xOpen& LIBRARY¶
DECLARE FUNCTION xRead& LIBRARY¶
DECLARE FUNCTION xWrite& LIBRARY¶
DECLARE FUNCTION xClose& LIBRARY¶
¶
DIM Matrix%(10,10)¶
¶
¶
WIDTH 70¶
¶
FOR i = 1 TO 10¶
  FOR j = 1 TO 10¶
     Matrix%(i,j) = RND*10¶
     PRINT Matrix%(i,j);" ";¶
  NEXT j¶
PRINT¶
NEXT i¶
PRINT¶
¶
¶
¶
File$ = "Matrix"+CHR$(0)¶
¶
Handle& = xOpen&(SADD(File$), 1006)¶
¶
  Amount& = xWrite&(Handle&, VARPTR(Matrix%(1,1)),
200&)¶
¶
CALL xClose(Handle&)¶
¶
FOR i = 1 TO 10¶
```

```
  FOR j = 1 TO 10¶
     Matrix%(i,j) = 0¶
NEXT j¶
NEXT i¶
¶
¶
Handle& = xOpen&(SADD(File$), 1005)¶
¶
  Amount& = xRead&(Handle&, VARPTR(Matrix%(1,1)),
200&)¶
¶
CALL xClose(Handle&)¶
¶
¶
FOR i = 1 TO 10¶
  FOR j = 1 TO 10¶
     PRINT Matrix%(i,j);" ";¶
  NEXT j¶
  PRINT¶
NEXT i¶
PRINT¶
```

Description du programme

Le programme remplit un tableau de 10 x 10 éléments avec des nombres aléatoires. La fonction xWrite& écrit ces données sur le disque. Le nombre 1006 dans la première ligne de la fonction xOpen() ouvre le fichier Matrix en écriture (1005 ouvre les fichiers en lecture).

Le programme remet le tableau à zéro. Ensuite, la fonction xRead() lit le fichier et place son contenu dans le tableau Matrix. Le contenu du tableau apparaît à l'écran et le programme s'achève.

Si vous en avez le temps et l'envie, répétez l'expérience avec un tableau composé d'un plus grand nombre d'éléments. Déterminez le nombre d'octets à lire et à écrire et introduisez cette valeur à la place de la valeur 200&. Remplacez les valeurs dans la boucle par les nouvelles valeurs et testez le programme ainsi modifié. Si votre curiosité ne s'arrête pas là, vous pouvez programmer le processus de sauvegarde dans sa totalité à l'aide des commandes BASIC et de xRead() et xWrite().

Les deux programmes présentés requièrent une mémoire à accès séquentiel. AmigaDOS permet à la fois un accès séquentiel

et direct aux fichiers. Les fichiers directs necéssitent des commandes qui dirigent le pointeur de fichier vers un emplacement spécifique. Si nous calculons cette position, le pointeur se déplace vers cet emplacement et y lit les données.

Concentrons notre attention sur les fichiers à accès direct. Entrez le programme suivant et exécutez le. Le programme tire ses informations des instructions DATA et écrits ces données dans le fichier addresses.obj:

```
'**********************************¶
'*             randomgen          *¶
'*                                *¶
'* génère un fichier direct       *¶
'* utilisable avec le programme   *¶
'* seekdemo de TRUCS ET ASTUCES   *¶
'*                                *¶
'*                                *¶
'**********************************¶
¶
OPEN "r",#1,"df1:Addresses.obj",100¶
FIELD #1,30 AS nam$,30 AS address$,20 AS city$,12 AS
phone$,8 AS comment$¶
CLS¶
PRINT:PRINT "Lecture des données et création d'un
fichier direct en cours."¶
PRINT:PRINT "Patientez...."¶
¶
viewandwrite:¶
¶
  READ xvalue,namedata$,addrdata$,citydata$,phondata$,
cmmtdata$¶
¶
IF xvalue-1 THEN¶
¶
 LSET     nam$=namedata$¶
 LSET     address$=addrdata$¶
 LSET     city$=citydata$¶
 LSET     phone$=phondata$¶
 LSET     comment$=cmmtdata$¶
         x=xvalue¶
     PUT #1,x¶
¶
 PRINT "Record "x" ("nam$") stored."¶
GOTO viewandwrite:¶
END IF¶
¶
CLOSE 1¶
PRINT:PRINT "   Fichier crée et fermé. Maintenant,
utilisez le programme seekdemo"¶
      PRINT "   pour lire chaque enregistrement."¶
```

```
END¶
¶
DATA 1,Jim Oldfield Jr.,5370 52nd Street SE, Grand
Rapids MI, 555-1212,printers¶
DATA 2,Doug D.R. Cotton,P.O. Box 2800,Sparta
MI,459-1212,good bbs¶
DATA 3,Jim D'Haem,5370 52nd Street SE,Grand Rapids
MI,957-4488,landlord¶
DATA 4,Dick Droste,P.O. Box 9999,Youpie
MI,555-1414,graphix¶
DATA 5,Gene Traas,RFD 2000,Hastings MI,999-1119,
proofer¶
DATA -1,,,,,¶
```

La longueur d'enregistrement du fichtest généré est de 100 octets. Entrez et exécutez le programme suivant. Ce programme ouvre le fichier addresses.obj et invite l'utilisateur à entrer un numéro d'enregistrement. Lorsque l'utilisateur entre un numéro de 1 à 5 puis enfonce la touche <Return>, le pointeur de fichier se déplace vers l'enregistrement voulu et l'affiche sur l'écran. La fonction Seek() détermine la position du pointeur.

```
'************************************¶
'*                                  *¶
'* Demo de Seek                     *¶
'* Accès direct au fichier par la   *¶
'*          fonction Seek           *¶
'*----------------------------------*¶
'*                                  *¶
'* Auteur : Wolf-Gideon Bleek       *¶
'* Date   : July 10, 1988           *¶
'* Version: 1.0                     *¶
'* Système d'expl.  : V1.2 & V1.3   *¶
'*                                  *¶
'************************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION xOpen& LIBRARY¶
DECLARE FUNCTION xRead& LIBRARY¶
DECLARE FUNCTION Seek&  LIBRARY¶
'                xClose¶
¶
DIM Memory%(100)¶
¶
WIDTH 30¶
¶
PRINT "N de l'enregistrement"¶
INPUT "(1-5):";no&¶
¶
File$ = "df1:Addresses.obj"+CHR$(0)¶
```

```
¶
Handle& = xOpen&(SADD(File$), 1005)¶
¶
  Position& = Seek&(Handle&, (no&-1)*100&, 0&)¶
  Amount& = xRead&(Handle&, VARPTR(Memory%(0)), 100&)¶
¶
CALL xClose(Handle&)¶
¶
FOR i = 0 TO 99¶
   PRINT CHR$(PEEK(VARPTR(Memory%(0))+i));¶
NEXT i¶
LIBRARY CLOSE¶
```

Accès au DOS par la fonction Lock.

Nous allons à présent nous intéresser à un autre type d'accès aux fichiers. Les méthodes décrites plus haut nous ont indiqué comment lire des données dans un fichier. Les données peuvent aussi être transférées. Il existe deux fonctions AmigaDOS pour réaliser cette opération.

La fonction Lock sélectionne le nom de fichier et le mode d'accès. Le mode d'accès équivaut à -2 si un accès en lecture est requis et à -1 dans le cas d'un accès en écriture. L'accès en lecture n'interdit pas l'exécution de plusieurs tâches sur un fichier ouvert. En revanche, lors d'une opération d'écriture, une seule tâche à la fois est autorisée.

UnLock() et xClose() sont des instructions similaires : l'accès au fichier et la mémoire réservée sont tous deux libérés.

Examinons le programme suivant. Il accède à n'importe quel fichier, sélectionne le mode d'accès puis le supprime. Nous nous servirons de ces informations plus tard à des fins de test.

```
'**********************************¶
'*                                *¶
'* Demo de Lock                   *¶
'* Accès au fichier par Lock      *¶
'* ------------------------------ *¶
'*                                *¶
'* Auteur : Wolf-Gideon Bleek     *¶
'* Date   : July 11, 1988         *¶
'* Version: 1.0                   *¶
'* Système d'expl. :  V1.2 & V1.3 *¶
'*                                *¶
'**********************************¶
¶
```

```
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Lock& LIBRARY¶
¶
¶
  PRINT "Entrez un nom de fichier S.V.P"¶
  PRINT¶
  INPUT "Nom du fichier";File$¶
  PRINT¶
¶
  File$  = File$ + CHR$(0)¶
  FileL& = Lock&(SADD(File$), -2&)¶
¶
  IF FileL&  0 THEN¶
    PRINT "Fichier trouvé, verrou positionné!"¶
    UnLock(FileL&)¶
  ELSE¶
    PRINT "Fichier non trouvé, verrou impossible!"¶
  END IF¶
```

2.3.2. Informations sur les fichiers, les répertoires et le disque

Lock fait bien plus qu'accéder aux arguments du fichier. Elles permet aussi d'accéder aux répertoires et aux disques.

Une fois que vous maitrisez la fonction Lock, vous maitrisez du même coup plusieurs autres fonctions qui s'exécutent concurremment. La première de ces fonctions que nous allons analyser est Examine(). Cette fonction tire des informations du fichier identifié par un Lock, comme vous l'avez probablement déjà vu dans les commandes CLI. Examine() necéssite une zone tampon où les données peuvent être transférées. La fonction AllocMem() ,se trouvant dans le fichier exec.bmap, crée cette zone mémoire.

Le programme suivant examine les données d'un fichier à l'aide de Lock et Examine() :

```
'************************************¶
'*                                  *¶
'* Demo de Exam                     *¶
'* Info. sur les fichiers par Examine *¶
'* --------------------------------   *¶
'*                                  *¶
'* Auteur : Wolf-Gideon Bleek       *¶
'* Date   : July 11, 1988           *¶
```

```
'* Version: 1.0                          *¶
'* System d'expl.  : V1.2 & V1.3        *¶
'*                                       *¶
'*****************************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Lock& LIBRARY¶
DECLARE FUNCTION Examine LIBRARY¶
¶
LIBRARY "T&A2:bmaps/exec.library"¶
¶
DECLARE FUNCTION AllocMem& LIBRARY¶
¶
Typ& = 2^16¶
¶
InfoBlock& = AllocMem&(300&, Typ&)¶
¶
IF InfoBlock& <> 0 THEN¶
¶
  PRINT "Entrez un nom de fichier S.V.P."¶
  PRINT¶
  INPUT "Nom du fichier:";File$¶
  PRINT¶
¶
  File$  = File$ + CHR$(0)¶
  FileL& = Lock&(SADD(File$), -2&)¶
¶
  IF FileL& <> 0 THEN¶
    PRINT "Lock set!"¶
    Status = Examine(FileL&, InfoBlock&)¶
    PRINT "DiskKey ";PEEKL(InfoBlock&)¶
    PRINT "DirType ";PEEKL(InfoBlock&+4)¶
    PRINT "Name    ";¶
    FOR i = 8 TO 38¶
       IF PEEK(InfoBlock&+i)<>0 THEN¶
         PRINT CHR$(PEEK(InfoBlock&+i));¶
       END IF¶
    NEXT i¶
¶
    UnLock(FileL&)¶
  ELSE¶
    PRINT "Verrou impossible!"¶
  END IF¶
¶
  CALL FreeMem(InfoBlock&, 300&)¶
¶
ELSE¶
¶
  PRINT "Pas de mémoire disponible--désolé!"¶
¶
END IF¶
```

Description du programme :

Après avoir ouvert les deux bibliothèques, notre programme recherche ,dans un premier temps, un bloc libre de mémoire pour y enregistrer les données ultèrieurement. Ensuite, le programme invite l'utilisateur à entrer un nom de fichier auquel un verrou sera affecté. Si tout fonctionne correctement, Examine() reçoit le bloc information du fichier et le place dans un segment de la mémoire tampon. Ensuite le programme affiche un certain nombre d'informations et le nom du fichier, puis il ferme toute les voies et les bibliothèques ouvertes et le programme s'achève.

La fonction ExNext() analyse l'enregistrement suivant d'un répertoire. Elle peut être également utilisée pour afficher une liste de répertoires. Le nom du verrou doit correspondre au nom du répertoire (exemple, libs,df1:, etc.). Le programme affiche les entrées du répertoire jusqu'à ce que la variable Status contienne un zéro.

```
'************************************¶
'*                                  *¶
'* Demo de ExNext                   *¶
'* Affichage d'un répertoire avec   *¶
'* Exnext                           *¶
'* -------------------------------- *¶
'*                                  *¶
'* Auteur : Wolf-Gideon Bleek       *¶
'* Date   : July 11, 1988           *¶
'* Version: 1.0                     *¶
'* Système d'expl  : V1.2 & V1.3    *¶
'*                                  *¶
'************************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Lock& LIBRARY¶
DECLARE FUNCTION Examine LIBRARY¶
DECLARE FUNCTION ExNext LIBRARY¶
¶
LIBRARY "T&A2:bmaps/exec.library"¶
¶
DECLARE FUNCTION AllocMem& LIBRARY¶
¶
Typ& = 2^16¶
¶
InfoBlock& = AllocMem&(300&, Typ&)¶
¶
IF InfoBlock& <> 0 THEN¶
¶
  PRINT "Entrez le nom du répertoire S.V.P."¶
```

```
  PRINT¶
  INPUT "Nom du disque:";Dir$¶
  PRINT¶
¶
  Dir$  = Dir$ + CHR$(0)¶
  DirL& = Lock&(SADD(Dir$), -2&)¶
¶
  IF DirL& <> 0 THEN¶
    Status = Examine(DirL&, InfoBlock&)¶
¶
    WHILE(Status<>0)¶
¶
      i = 8¶
¶
      WHILE(PEEK(InfoBlock&+i)<>0)¶
¶
        PRINT CHR$(PEEK(InfoBlock&+i));¶
        i = i + 1¶
¶
      WEND¶
¶
      PRINT¶
      Status = ExNext(DirL&, InfoBlock&)¶
¶
    WEND¶
¶
    UnLock(DirL&)¶
¶
    PRINT "Non trouvé, verrou impossible!"¶
¶
  END IF¶
¶
  CALL FreeMem(InfoBlock&, 300&)¶
¶
ELSE¶
¶
  PRINT "Pas de mémoire disponible--désolé!"¶
¶
END IF¶
```

Description du programme

Le fonctionnement de ce programme est analogue à celui du programme précédent, à l'exception près que la boucle principale de ce programme est plus sophistiquée. Cette boucle permet de lire facilement le premier enregistrement du répertoire à l'aide de Examine. Une fois le nom du répertoire entré, le programme active le verrou.

Puis, le programme recherche les enregsitrements suivants au moyen de ExNext(). Il affiche le nom du répertoire suivant et la boucle se poursuit aussi longtemps que la variable Status renvoie une valeur différente de zéro. Si le programme rencontre une valeur nulle dans la variable Status, le programme gènère le message d'erreur NO_MORE_ENTRIES et s'interrompt.

AmigaDOS fournit trois autres fonctions de gestion de répertoires. La fonction ParentDir() détermine si un répertoire donné contient un sous-répertoire. Ce programme nécessite un verrou sur le répertoire en cours et renvoie soit un verrou vers le sous répertoire soit une valeur nulle (si le programme se trouve déjà dans le répertoire principal). Le programme ci-dessous utilise cette fonction et affiche le niveau de répertoire courant.

```
'*********************************¶
'* ParentDir                      *¶
'*     Liste d'un niveau de       *¶
'*     répertoire                 *¶
'* ------------------------------ *¶
'*                                *¶
'* Auteur : Wolf-Gideon Bleek     *¶
'* Date   : 11 juillet 1988       *¶
'* Version: 1.0                   *¶
'* Système d'expl  : V1.2 & V1.3*¶
'*                                *¶
'*********************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Lock& LIBRARY¶
DECLARE FUNCTION ParentDir& LIBRARY¶
¶
  PRINT "Entrez le nom du répertoire S.V.P."¶
  PRINT¶
  INPUT "Nom:";Dir$¶
  PRINT¶
¶
  Level = 0¶
¶
  Dir$  = Dir$ + CHR$(0)¶
  DirL& = Lock&(SADD(Dir$), -2&)¶
¶
  IF DirL& <> 0 THEN¶
¶
    WHILE(DirL& <> 0)¶
¶
      DirL& = ParentDir&(DirL&)¶
      Level = Level + 1¶
```

```
¶
    WEND¶
¶
    UnLock(DirL&)¶
¶
    IF Level = 1 THEN¶
      PRINT ">> Répertoire principal! <<"¶
    END IF¶
¶
    PRINT "Niveau: ";Level¶
¶
  ELSE¶
¶
    PRINT "Verrou impossible!"¶
¶
  END IF¶
```

Description du programme

Le programme parcourt les niveaux de répertoire jusqu'à ce qu'une valeur nulle soit renvoyée, ce qui se produit lorsque le programme principal est atteint. L'invocation de ParentDir() renvoie le niveau du répertoire. Vous pouvez indiquer plusieurs répertoires dans le chemin pour trouver différents niveaux.

Il nous reste à exposer deux fonctions qui s'appliquent directement aux répertoires. La première, CurrentDir() est semblable à la commande CHDIR d'AmigaBASIC. L'utilisateur définit un verrou sur le répertoire et accède à cette fonction. L'ancien verrou est renvoyé jusqu'à l'obtention du répertoire courant. De cette façon, vous pouvez redéfinir les chemins de répertoire à n'importe quel moment.

Il est aussi possible d'accéder aux répertoires à partir de dos.library. La dernière fonction requiert simplement le nom du nouveau répertoire. Le répertoire sera considéré comme une sous-répertoire du répertoire courant. Par conséquent, vous devez utiliser CurrentDir() avant d'accéder à un répertoire quelconque. Le résultat est équivalent à un verrou sur le nouveau répertoire.

Nous savons à présent comment accéder aux répertoires et obtenir des informations. Il nous reste à rechercher des informations sur le disque. AmigaDOS possède une troisième fonction de répertoire appelée Info().

Cette fonction charge les informations relatives au disque et les place dans une mémoire tampon. Ceci nous permet d'afficher et de lire très simplement des données essentielles du disque.

```
'*********************************¶
'*                               *¶
'* INFODEMO                      *¶
'* Info. sur le disque par Info  *¶
'* ----------------------------  *¶
'*                               *¶
'* Auteur : Wolf-Gideon Bleek    *¶
'* Date   : July 11,1988         *¶
'* Version: 1.0                  *¶
'* Système d'expl  : V1.2 & V1.3 *¶
'*                               *¶
'*********************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Lock& LIBRARY¶
DECLARE FUNCTION Info LIBRARY¶
¶
LIBRARY "T&A2:bmaps/exec.library"¶
¶
DECLARE FUNCTION AllocMem& LIBRARY¶
¶
typ& = 2^16¶
¶
Info& = AllocMem&(300&, typ&)¶
¶
IF Info& <> 0 THEN¶
¶
  PRINT "Entrez un nom de périphérique S.V.P."¶
  PRINT¶
  INPUT "Nom:";Dev$¶
  PRINT¶
¶
  Dev$  = Dev$ + CHR$(0)¶
  DevL& = Lock&(SADD(Dev$), -2&)¶
¶
  IF DevL& <> 0 THEN¶
¶
    Status = Info(DevL&, Info&)¶
¶
    i = 8¶
    WHILE(PEEK(Info&+i)<>0)¶
      PRINT CHR$(PEEK(Info&+i));¶
    WEND¶
¶
    PRINT¶
¶
```

```
    PRINT "Erreurs logicielles: ";PEEKL(Info&)¶
    PRINT "NO de l'unité: ";PEEKL(Info&+4)¶
    PRINT "Etat du disque: "; : st& = PEEKL(Info&+8)¶
    IF st& = 80 THEN PRINT "Protégé en écriture"¶
    IF st& = 81 THEN PRINT "Validé"¶
    IF st& = 82 THEN PRINT "Disque prêt"¶
¶
    PRINT¶
    PRINT "Nb de blocs  : ";PEEKL(Info&+12)¶
    PRINT "Nb utilisé   : ";PEEKL(Info&+16)¶
    PRINT "Octets/bloc: ";PEEKL(Info&+20)¶
    PRINT "Type du disque   : "; : typ& = PEEKL
(Info&+24)¶
    FOR i = 3 TO 0 STEP -1¶
      PRINT CHR$((typ& / 256^i) AND 255);¶
    NEXT i¶
¶
    PRINT¶
¶
    UnLock(DevL&)¶
¶
  ELSE¶
¶
    PRINT "Verrou impossible!"¶
¶
  END IF¶
¶
  CALL FreeMem(Info&, 300&)¶
¶
ELSE¶
¶
  PRINT "Pas de mémoire disponible--désolé!"¶
¶
END IF¶
```

Description du programme

Après avoir placé un verrou sur une entrée du disque, la fonction Info() lit les informations. Ces données sont ensuite affichées à l'écran. Puis, l'accès au disque prend fin et le programme libère la mémoire allouée.

2.3.3. Gestion directe des fichiers

Ce chapitre est consacré à quatre fonctions parmi lesquelles vous reconnaitrez trois fonctions du CLI. Nous exposons ,ici, la façon dont vous pouvez utiliser ces fonctions sous AmigaBASIC.

Commencez par créer trois programmes de test simples sous AmigaBASIC puis sauvegardez les dans un même répertoire (l'instruction REM ou PRINT fera l'affaire). Baptisez ces fichiers de test EffaceTest, Mon_Ancien_Nom et Protège_Moi. Entrez les trois programmes de démonstration qui suivent et sauvegardez les dans le même répertoire que les trois programmes qui viennent d'être créés (de préférence, le répertoire principal).

La fonction DeleteFile() est totalement compatible avec la commande Delete du CLI. Si l'on entre le fichier sous la forme d'une chaîne de caractères, celui-ci est effacé. Il faut souligner que DeleteFile() ne supprime pas automatiquement le fichier .info correspondant. Ce dernier doit être effacé séparément. Voici un court programme de démonstration qui supprime le programme BASIC EffaceTest ainsi que le fichier EffaceTest.info :

```
' demo de DeleteFile¶
' Efface le fichier BASIC intitulé EffaceTest¶
' et son fichier .info¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION DeleteFile LIBRARY¶
¶
file$ = "EffaceTest" + CHR$(0)¶
status = DeleteFile(SADD(file$))¶
¶
file$ = "EffaceTest.info" + CHR$(0)¶
status = DeleteFile(SADD(file$))¶
```

La fonction Rename() est entièrement compatible avec la commande Rename du CLI. L'utilisateur entre deux noms de fichier : l'ancien et le nouveau. Voici un court programme de démonstration qui renomme le programme.Mon_Ancien_Nom et son fichier .info en Le_Nouveau_Truc :

```
' Demo de Rename¶
' Renomme le programme BASIC intitulé¶
' Mon_Ancien_Nom to Le_Nouveau_Truc¶
' et ses fichiers .info¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
DECLARE FUNCTION Rename LIBRARY¶
¶
OldFile$="Mon_Ancien_Nom"+CHR$(0)¶
NewFile$="Le_Nouveau_Truc"+CHR$(0)¶
OldInfo$="Mon_Ancien_Nom.info"+CHR$(0)¶
```

```
NewInfo$="Le_Nouveau_Truc.info"+CHR$(0)¶
¶
Status=Rename(SADD(OldFile$), SADD(NewFile$))¶
Status=Rename(SADD(OldInfo$), SADD(NewInfo$))¶
```

La dernière fonction SetProtection() valide et invalide les bits de protection des fichiers de la version 1.2 (quatre bits de protection) et de la version 1.3 (huit bits de protection). L'ordre des bits varie en fonction du fichier. Ainsi, vous pouvez protéger vos fichier contre l'effacement. Voici un court programme de démonstration (changez les noms de fichier pour les adapter à vos besoins)

```
' Demo de SetProtection¶
' Positionne les bits de protection du fichier
intitulé¶
' Protège_Moi¶
LIBRARY "T&A2:bmaps/dos.library"¶
DECLARE FUNCTION SetProtection LIBRARY¶
¶
File$ = "Protège_Moi"+CHR$(0)¶
Mask& = 4+8¶
status = SetProtection(SADD(File$), Mask&)¶
```

La variable mask& peut être subdivisée comme suit :

bit#	Valeur du bit égale à 1
0	Le fichier ne peut être effacé
1	Le fichier ne peut être exécuté (FFS seulement)
2	Le fichier ne peut être écrasé (FFS seulement)
3	Le fichier ne peut être lu (FFS seulement)
4	Le fichier reste inchangé après une copie
5	Le fichier programme peut devenir résident
6	Le fichier est un fichier script, et peut être lancé sans Execute
7	Le fichier ne peut être visualisé (n'existe pas)

2.4. Bibliothèques et applications

Nous vous avons présenté, jusque là, de courts exemples de fonctions DOS accessibles sous AmigaBASIC. Ces programmes sont suffisants pour vous donner un avant-goût du sujet. Nous conclurons ce chapitre avec un programme plus long qui peut s'avérer extrêmement utile pour un programmeur en BASIC.

Vous trouverez certainement de nombreuses applications à ce programme. Il vous permettra de réaliser des listings d'archivage des répertoire de disque. Il débute par la création de listes de répertoire en vue d'une impression. Une fois la liste de répertoire générée, vous pouvez l'afficher et dans une certaine mesure la manipuler, et même la trier par ordre alphabétique.

2.4.1. Le gestionnaire de répertoire

L'objectif de ce programme est de vous permetre de gérer des répertoires, de les lire et de les afficher sous la forme d'un tableau. Cette liste peut être triée par ordre alphabétique des fichiers et des répertoires.

Ce système convient parfaitement aux programmes et aux textes constitués de nombreux fichiers qu'il est généralement malaisé de lire à l'aide d'une commande normale du fait de la rapidité de leur défilement à l'écran (par exemple, files dans AmigaBASIC). Dresser une liste de répertoires facilite la recherche d'un fichier et convertir un répertoire en un fichier vous permet de le lire à partir d'un traitement de texte.

```
'*************************************¶
'* dirmanager                        *¶
'*   Gestionnaire de répertoire      *¶
'*   sous AmigaBASIC                 *¶
'* --------------------------------- *¶
'*                                   *¶
'* Auteur : Wolf-Gideon Bleek        *¶
'* Date   : July 11, 1988            *¶
'* Version: 1.0                      *¶
'* Système d'expl. : V1.2 & V1.3     *¶
'*                                   *¶
'*************************************¶
¶
LIBRARY "T&A2:bmaps/dos.library"¶
¶
```

```
DECLARE FUNCTION Lock& LIBRARY¶
DECLARE FUNCTION Examine LIBRARY¶
DECLARE FUNCTION ExNext LIBRARY¶
¶
LIBRARY "T&A2:bmaps/exec.library"¶
¶
DECLARE FUNCTION AllocMem& LIBRARY¶
¶
Info&     = 0¶
¶
Typ&      = 2^16¶
MaxSize&  = 300¶
¶
Suffix$   = "no ending"¶
¶
DIM Dir$(200), File$(200), Size$(200)¶
¶
Info& = AllocMem&(MaxSize&, Typ&)¶
OPEN "ram:temporary" FOR OUTPUT AS 255¶
¶
OPEN "RAM:dir-list" FOR APPEND AS 1¶
CLOSE 1¶
¶
MENU 1,0,1,"Données  "¶
MENU 1,1,1,"Charger  "¶
MENU 1,2,1,"Sauver   "¶
MENU 1,3,1,"Quitter  "¶
¶
MENU 2,0,1,"Répertoire "¶
MENU 2,1,1,"Achever    "¶
MENU 2,2,1,"Trier      "¶
MENU 2,3,1,"Afficher   "¶
MENU 2,4,1,"Effacer    "¶
¶
MENU 3,0,1,"Règlages"¶
MENU 3,1,1,"Extensions Filtre"¶
¶
MENU 4,0,0,""¶
¶
ON MENU GOSUB Selects¶
¶
MENU ON¶
¶
WHILE 1¶
    IF FileMemory& <> 0 THEN¶
    FreeMem(FileMemory&, Length&)¶
  END IF¶
¶
SLEEP¶
WEND¶
¶
Selects:¶
```

```
  HM = MENU(0)¶
  MP = MENU(1)¶
¶
  FOR i = 1 TO 3¶
    MENU i, 0, 0¶
  NEXT i¶
¶
  IF HM = 1 AND MP = 3 THEN¶
    MENU RESET¶
    CALL FreeMem(Info&, MaxSize&)¶
    CLOSE 255¶
    LIBRARY CLOSE¶
    END¶
  END IF¶
¶
  IF HM = 1 AND MP = 1 THEN GOSUB LoadIt¶
  IF HM = 1 AND MP = 2 THEN GOSUB SaveData¶
¶
  IF HM = 2 AND MP = 1 THEN GOSUB GetDir¶
  IF HM = 2 AND MP = 2 THEN GOSUB Sort¶
  IF HM = 2 AND MP = 3 THEN GOSUB DisplayIt¶
  IF HM = 2 AND MP = 4 THEN GOSUB DeleteIt¶
¶
  IF HM = 3 AND MP = 1 THEN GOSUB NewSuffix¶
¶
  PRINT¶
  PRINT "OK"¶
  PRINT¶
¶
  FOR i = 1 TO 3¶
    MENU i,0,1¶
  NEXT i¶
¶
  RETURN¶
¶
LoadIt:¶
  PRINT "CHARGEMENT du fichier créé par dirmanager"¶
  PRINT¶
¶
  PRINT "Entrez le nom du programme S.V.P."¶
  PRINT¶
  INPUT "Nom du fichier:";SourceFile$¶
¶
  IF SourceFile$ = "" THEN RETURN¶
¶
  PRINT¶
  PRINT "Chargement du fichier en cours..."¶
¶
  OPEN "RAM:dir-list" FOR OUTPUT AS 2¶
  OPEN SourceFile$ FOR INPUT AS 1¶
¶
  WHILE(EOF(1)=0)¶
¶
```

```
    INPUT#1, Dir$, File$, Size$¶
    PRINT#2, Dir$¶
    PRINT#2, File$¶
    PRINT#2, Size$¶
¶
  WEND¶
¶
  CLOSE 1¶
  CLOSE 2¶
¶
  RETURN¶
¶
SaveData:¶
  PRINT "SAUVEGARDE du fichier créé par dirmanager"¶
  PRINT¶
¶
  PRINT "Entrez le nom du fichier S.V.P."¶
  PRINT¶
  INPUT "Nom du fichier:";DestFile$¶
¶
  IF DestFile$ = "" THEN RETURN¶
¶
  PRINT¶
  PRINT "Sauvegarde du fichier en cours..."¶
¶
  OPEN "RAM:dir-list" FOR INPUT AS 1¶
  OPEN DestFile$ FOR OUTPUT AS 2¶
¶
  WHILE(EOF(1)=0)¶
¶
    INPUT#1, Dir$, File$, Size$¶
    PRINT#2, Dir$¶
    PRINT#2, File$¶
    PRINT#2, Size$¶
¶
  WEND¶
¶
  CLOSE 1¶
  CLOSE 2¶
¶
  RETURN¶
¶
GetDir:¶
  PRINT "Chargement du répertoire"¶
  PRINT¶
¶
  PRINT "Entrez le chemin du répertoire S.V.P."¶
  PRINT¶
  INPUT "Nom du répertoire:";Dir$¶
¶
  IF Dir$ = "" THEN RETURN¶
¶
```

```
  PRINT¶
  PRINT "Lecture du répertoire en cours..."¶
  PRINT¶
¶
  Dir$  = Dir$ + CHR$(0)¶
  DirLk& = Lock&(SADD(Dir$), -2&)¶
¶
  IF DirLk& <> 0 THEN¶
¶
    OPEN "RAM:dir-list" FOR APPEND AS 1¶
¶
    Status = Examine(DirLk&, Info&)¶
    PRINT "Directory: ";¶
    CALL Display(Info&, 255, "")¶
    Directories$ = File$¶
¶
    Status = ExNext(DirLk&, Info&)¶
¶
    WHILE(Status<>0)¶
¶
      CALL Display(Info&, 1, Directories$)¶
      Status = ExNext(DirLk&, Info&)¶
¶
    WEND¶
¶
    CLOSE 1¶
¶
  ELSE¶
¶
    PRINT "Répertoire non trouvé!"¶
¶
  END IF¶
¶
  RETURN¶
¶
Sort:¶
¶
  PRINT "Tri du répertoire"¶
  PRINT¶
¶
  OPEN "ram:dir-list" FOR INPUT AS 1¶
¶
  z = 0¶
¶
  WHILE(EOF(1)=0)¶
¶
    INPUT#1, Dir$(z), File$(z), Size$(z)¶
    z = z + 1¶
¶
  WEND¶
¶
  CLOSE 1¶
¶
```

```
  FOR i = 0 TO z - 2¶
    k = i¶
    FOR j = i TO z - 1¶
      IF Dir$(j)+File$(j)<Dir$(k)+File$(k) THEN k = j¶
    NEXT j¶
    SWAP Dir$(i), Dir$(k)¶
    SWAP File$(i), File$(k)¶
    SWAP Size$(i), Size$(k)¶
  NEXT i¶
¶
  OPEN "ram:dir-list" FOR OUTPUT AS 1¶
¶
  FOR i = 0 TO z - 1¶
    PRINT#1, Dir$(i)¶
    PRINT#1, File$(i)¶
    PRINT#1, Size$(i)¶
  NEXT i¶
¶
  CLOSE 1¶
¶
  RETURN¶
¶
DisplayIt:¶
¶
  PRINT "Listage du répertoire"¶
  PRINT¶
¶
  OPEN "ram:dir-list" FOR INPUT AS 1¶
¶
  WHILE(EOF(1)=0)¶
¶
    INPUT#1, Dir$, File$, Size$¶
    PRINT Dir$;TAB(40);File$;TAB(70);Size$¶
¶
  WEND¶
¶
  CLOSE 1¶
¶
  RETURN¶
¶
DeleteIt:¶
¶
  PRINT "Suppression de dir-list dans la RAM"¶
  PRINT¶
  PRINT "Voulez vous réellement effacer toutes les
données?"¶
¶
  INPUT "Y/N<Return>: ";a$¶
¶
  IF a$<>"Y" THEN RETURN¶
¶
  KILL "ram:dir-list"¶
¶
```

```
  PRINT¶
  PRINT "Répertoire effacé"¶
¶
  OPEN "ram:dir-list" FOR APPEND AS 1¶
  CLOSE 1¶
¶
  RETURN¶
¶
NewSuffix:¶
¶
  PRINT "Suppression d'extensions de fichiers
définie par l'utilisateur"¶
¶
  PRINT¶
¶
  PRINT "Entrez une extension S.V.P (ex., .ext)"¶
  PRINT "ou appuyez sur <Return> pour afficher tous
les fichiers."¶
  PRINT¶
¶
  INPUT "Extension: ";Suffix$¶
¶
  IF Suffix$ = "" THEN Suffix$ = "No ending"¶
¶
  RETURN¶
¶
SUB Display(Mem&, Ch%, Dir$) STATIC¶
SHARED File$, Suffix$¶
¶
  i = 8 : File$=""¶
¶
  WHILE(PEEK(Mem&+i)<>0)¶
    File$ = File$ + CHR$(PEEK(Mem&+i))¶
    i = i + 1¶
¶
  WEND¶
¶
  IF RIGHT$(UCASE$(File$), LEN(Suffix$)) = UCASE$
(Suffix$) THEN EXIT SUB¶
¶
  PRINT File$;TAB(35);¶
  PRINT#Ch%, Dir$¶
  PRINT#Ch%, File$¶
  IF PEEKL(Mem&+4)<0 THEN¶
    PRINT PEEKL(Mem&+124);TAB(45);¶
    PRINT#Ch%, PEEKL(Mem&+124)¶
  ELSE¶
    PRINT " DIR";TAB(45);¶
    PRINT#Ch%, "DIR"¶
  END IF¶
¶
  PRINT¶
END SUB¶
```

Chapitre 3

Le langage machine

AmigaBASIC est un bon langage de programmation, mais n'est certainement pas le plus rapide. Le meilleur langage de programmation est sans conteste le langage machine puisque la vitesse d'exécution de ses instructions est mille fois supérieure à celle des commandes en BASIC. Le langage machine permet d'accéder à toutes les fonctions de l'ordinateur et d'invoquer à tout moment les routines système auxquelles BASIC ne permet pas d'accéder. Il est vrai que le langage machine est ardu, mais on peut apprendre à programmer relativement vite sur Amiga avec un bon assembleur. Profimat est un excellent assembleur pour les débutants.

Disons deux mots du langage C dont la rapidité d'exécution est dix fois moindre que celle du langage machine. Presque toutes les opérations réalisables en langage machine le sont aussi en C. Malheureusement, en langage C, l'appel de certaines routines exceptionnelles requiert une assez longue programmation. Autant éviter toute cette peine et programmer directement en langage machine. De plus il est bien connu que le C n'utilise pas toutes les instructions du 68000, ou du moins pas toujours de manière efficace.

Vous ne connaissez peut être rien au langage machine. Cela ne pose aucun problème, car ce chapitre n'exige aucune connaissance particulière. Toutefois, si vous désirez devenir tout à fait opérationnel, nous vous conseillons de vous atteler à l'apprentissage du langage machine 68000. Procurez vous un ouvrage de base sur le sujet (Le livre du langage machine Amiga paru aux Editions Micro Application est un bon ouvrage sur le code

machine 68000 de l'Amiga) et potassez le. Nous le répétons encore une fois: ces connaissances ne sont pas indispensables pour aborder ce chapitre _ Les codes y sont expliqués avec toute la minutie possible.

Le présent chapitre comprend de nombreux programmes et routines en langage machine qui vous seront utiles. Nous y avons même ajouté un programme conçu pour éliminer les virus (un véritable problème dans la communauté Amiga).

Les routines présentées ont été écrites avec l'assembleur DevPac de HiSoft. Pour ceux qui ne possèdent pas d'assembleur, nous avons inclus des chargeurs BASIC aux programmes chaque fois que possible.

3.1. La division par zéro

La division d'une valeur par zéro est l'une des causes les plus fréquentes de génération d'une Méditation du Gourou. Avant d'en arriver à l'exposé des solutions, observons ce qui se produit pendant et aprés une division par zéro.

Le processeur génère une exception. Le registre d'état du 68000 est transféré dans une mémoire tampon. Le processeur passe automatiquement en mode Superviseur. Le bit de Trace est effacé pour désactiver le mode pas à pas. Le compteur ordinal, le registre d'état, le registre instructions (et son état lorsque l'erreur s'est produite), l'adresse d'accès et le mot Superstate (qui indique l'état du processeur) sont placés sur la pile Superviseur.

Aprés une routine d'exception, le processeur poursuit l'exécution du programme qui a généré l'erreur; l'instruction RTE (ReTurn from Exception) est exécutée. Le vecteur d'exception apparaît au niveau de RTE. La commande SetAlert du CLI constitue une variante de cela. Cette commande attend qu'un drapeau(flag) d'erreur soit positionné puis elle détermine en fonction de l'état du mot Superstate si un RTE ou un Gourou doit être déclenché.

Nous aimerions clarifier le point suivant : lorsqu'un programme rencontre un dénominateur dont la valeur est si petite qu'elle ne peut être représentée par un mot (les commandes DIVU et DIVS du 68000 ne traitent que les mots), celle-ci est arrondie à zéro. Là est le problème : le 68000 ne peut pas diviser par zéro.

L'instruction RTE nous renvoie à l'adresse du programme située juste aprés la commande à l'origine de l'erreur. Parfait, direz vous. Eh bien vous avez tort! Réfléchissez à la chose suivante : Que se passe-t-il lors d'une division par un nombre très petit, par exemple .00001? La division peut donner comme résultat un nombre assez grand.

Le registre, dans lequel le résultat est enregistré, contient à présent une valeur très petite. Par conséquent, tous les calculs ultérieurs seront nécessairement faux dès qu'une nouvelle routine d'exception sera créée. Le programme ne renvoie que des résultats aberrants dont l'utilisateur n'a que faire. Supposons que le dénominateur prenne une valeur extrêmement petite au lieu de zéro. La division redevient alors possible. Nous pouvons substituer "infiniment petit" par :

```
              1
-------------------------------
  Dénominateur Infiniment_grand
```

La division par ce dénominateur nous permet de tirer parti de la bonne vieille règle qui dit que pour diviser par une fraction, il faut multiplier par l'inverse.

```
Compteur = x¶
Dénominateur inverse = infiniment_grand¶
¶
Résultat = Compteur * Dénominateur inverse¶
       = x * infiniment_grand¶
       = infiniment_grand¶
```

Il faut attribuer à infiniment_grand la plus grande valeur que les commandes DIVS et DIVU puissent traiter de façon à obtenir des calculs corrects malgré l'exception, et écrire une routine d'exception qui enregistre ces valeurs dans les registres correspondants avant l'exécution d'une instruction RTE.

Pour cette raison, il est nécessaire de savoir quelle commande est à l'origine de l'exception, la valeur la plus élevée possible étant $FFFF pour DIVU et $FFF pour DIVS. En outre, il est indispensable de déterminer dans quel registre de données cette valeur doit être transférée. Le numéro du registre de données est trouvé directement dans le code opération comme le montre le tableau suivant :

Commande	Bits du code de la commande															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVU	1	0	0	0	x	x	x	0	1	1	y	y	y	y	y	y
DIVS	1	0	0	0	x	x	x	1	1	1	y	y	y	y	y	y

Les bits 9 à 11 (x) renvoient le numéro du registre de données où le résultat est stocké. Les bits 0 à 5 donnent le type d'adressage, que vous n'avez pas besoin de connaître ici. Voici la version du programme en langage machine :

```
;Division par zéro - Gestionnaire; par SM'88¶
¶
Init_Trap:              ;Installation du gestionnaire¶
  Move.l 4,A6              ;ExecBase dans A6¶
  Move.l #Div_End-Div,D0  ;Handler-length jusqu'à D0¶
¶
  MoveQ  #1,D1             ;MEMF_Public¶
  Jsr    -198(a6)          ;appel AllocMem¶
  Move.l  D0,New_Trap      ;Lecture de l'adresse¶
¶
  Beq.s  Init_End          ;Fin, si erreur¶
  Move.l D0,A1             ;Adresse dans A1¶
  Lea    Div,A0            ;CodeStart-Address¶
¶
  MoveQ  #(Div_End-Div)-1,D0  ;CodeLength-1¶
Copy_Code:                 ;Copie du gestionnaire¶
  Move.b (A0)+,(A1)+       ;Copie de l'octet¶
  DBra   D0,Copy_Code      ;Octet suivant¶
  Move.l New_Trap,20       ;Nouveau Trap-Vector¶
¶
¶
¶
Init_End:              ;Adieu¶
  MoveQ  #0,D0         ;No = 0 !!!¶
  Divu   D0,D1         ;En avant, gâche ma journée¶
  rts                  ;Fin¶
¶
New_Trap:              ;Nouveau Trap-Vector¶
  dc.l   0             ;1 Long¶
Div:                   ;Gest. de division par 0¶
 MoveM.l D0-A6,-(sp) ;Tous les registres (s.u.)¶
  Move.l  62(sp),A0    ;Lire compt. ordinal à partir
de la pile¶
  Move.w  -2(A0),D0    ;Lire dernière commande¶
  Move.l  #$ffff,D1    ;Grande valeur pour DIVU¶
¶
  Btst    #8,D0        ;C'était une instruction DIVU?¶
  Beq.s   GoOn         ;alors continuer¶
  Move.l  #$7fff,D1    ;Sinon valeur de  Divs¶
```

```
¶
GoOn:                   ;Transfert des registres de
données¶
  Lsr.w   #7,D0         ;Défilement commande au niveau
des bits¶
  AndI.l  #28,D0        ;Ignorer registre¶
¶
  Move.l  D1,0(sp,d0.l);Modifier registres et piles¶
  MoveM.l (sp)+,D0-A6 ;Charger registres modifiés¶
  Rte                   ;Return from Exception¶
Div_End:                ;Etiquette: SizeOf¶
¶
  End¶
```

Vous vous demandez probablement de quelle façon la nouvelle valeur est transférée dans le registre de données. Une fois la totalité du registre stocké sur la pile, qui enregistre toujours l'adresse la plus haute en premier, les autres registres sont utilisés par ordre décroissant (se reporter à la troisième ligne du programme).Pour accéder à la pile, on utilise les numéros des registres multipliés par 4 comme déplacement . Si quelque chose ne fonctionne pas correctement, le système génère un Gourou.

Nous avons généré une Exception de division par zéro aprés avoir installé le nouveau vecteur d'interruption (Trap_Vector). La division par zéro a lieu dans la ligne 20 du programme.

Voici une courte routine en BASIC qui enregistre le programme sous la forme d'une commande CLI (vous devrez appeler cette commande à l'aide de votre séquence de démarrage).:

```
OPEN "sys:c/DIVZERO" FOR OUTPUT AS 1¶
FOR i=1 TO  176¶
 READ a$¶
 a%=VAL("&H"+a$)¶
 PRINT #1,CHR$(a%);¶
NEXT¶
CLOSE 1¶
KILL "sys:c/DIVZERO.info"¶
datas:¶
DATA 0,0,3,F3,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0¶
DATA 40,0,0,1B,0,0,3,E9,0,0,0,1B,2C,78,0,4,20,3C,0,0¶
DATA 0,30,72,1,4E,AE,FF,3A,23,C0,0,0,0,36,67,18,22,40,41,F9¶
DATA 0,0,0,3A,70,2F,12,D8,51,C8,FF,FC,21,F9,0,0,0,36,0,14¶
DATA 70,0,82,C0,4E,75,0,0,0,0,48,E7,FF,FE,20,6F,0,3E,30,28¶
DATA FF,FE,22,3C,0,0,FF,FF,8,0,0,8,67,6,22,3C,0,0,7F,FF¶
DATA EE,48,2,80,0,0,0,1C,2F,81,8,0,4C,DF,7F,FF,4E,73,0,0¶
DATA 0,0,3,EC,0,0,0,3,0,0,0,0,0,0,12,0,0,0,1C¶
DATA 0,0,0,2A,0,0,0,0,0,0,3,F2,0,0,3,F2¶
```

3.2. Alerte aux virus

Dans les milieux de l'informatique, la question des virus revient souvent au centre des discussions. D'aucuns disent que les virus informatiques n'existent pas, que ce ne sont que des bruits répandus par les médias. Laissons les autres en débattre. Pour notre part, nous croyons à leur existence.

Les virus informatiques se répandent à une allure vertigineuse. Sur Amiga, ce sont les virus SCA et Byte bandit que l'on rencontre le plus fréquemment. L'on serait enclin à penser que l'utilisation de logiciels vendus dans le commerce nous met hors d'atteinte. Ceci n'est plus vrai aujourd'hui. Nous sommes confronté à un problème dont il ne faut pas sous-estimer la gravité.

Qui est responsable ?

Face au problème des virus, la première question qui vient à l'esprit est de savoir "Qui en sont les auteurs?" A l'origine, les virus proviennent des pirates de logiciels qui pénètrent au coeur d'un programme commercialisé et de temps à autre y implantent un virus. Les copies piratées répandent alors les virus qui quelquefois reviennent même s'introduire dans les programmes originaux du fabricant. Ces derniers quittent alors l'usine porteurs d'un virus. Quelques fabricants de jeux ont connu ce genre de mésaventures. C'est le cas d'un fabricant de logiciels (conçus pour un autre type d'ordinateur muni d'un processeur 68000) reconnu dans la profession, qui a eu la mauvaise surprise de constater que les dirigeants d'un magazine avaient implanté un virus sur un logiciel de démonstration assez coûteux et perfectionné.

Comment un virus se propage-t-il ? Généralement, le virus se cache dans le bloc d'initialisation d'une disquette. Lorsque vous initialisez votre ordinateur avec la disquette en question, le système d'exploitation charge les secteurs d'initialisation (les deux premiers secteurs d'un disque). Ces derniers contiennent en général la routine d'initialisation de la bibliothèque DOS. La commande Install écrit cette routine sur les disques pour les rendre initialisables. Le système d'exploitation se branche directement sur la routine d'initialisation, ce que précisément le virus attend.

Le virus se copie lui-même dans une zone de mémoire, change de vecteurs de système et se dissémine dans la routine d'initialisation du DOS. Il s'installe ainsi dans le système sans se faire remarquer. Si vous insérez une nouvelle disquette d'initialisation dans le lecteur, le virus se copie lui même dans le bloc d'initialisation. Un autre type de virus réalise la même opération au cours d'une réinitialisation.

Malheureusement, de nombreux programmes démarrent à l'aide d'un chargeur situé dans le bloc d'initialisation que le virus écrase tout simplement, détruisant ainsi le disque. De plus, les ordinateurs contaminés subissent de nombreuses interruptions causées par le virus. Ceci est décelable ,en premier lieu, lorsque le virus décide qu'il a contaminé suffisamment de disques (il garde une trace du nombre de disques contaminés).

Au moment où nous écrivons cet ouvrage, le seul antidote connu est la commande Install qui écrase le bloc d'initialisation contaminé. Certains virus sont capables de reconnaître l'emploi d'Install. Lorsque cette commande commence à écrire sur le bloc d'initialisation, le virus positionne un drapeau quelque part au cours de la procédure d'écriture et reformate le disque. Les ordinateurs attaqués par ce type de virus sont généralement incurables.

3.2.1. L'arme suprême contre les virus

Le programme suivant doit être enregistré dans la séquence de démarrage de toutes les disquettes d'initialisation que vous possédez. Il examine les vecteurs du système que le virus peut utiliser et les efface. La totalité du disque d'initialisation doit être reconfiguré à l'aide de la commande Install. La neutralisation du programme de virus en mémoire empêche son écriture sur le disque après Install.

```
 ;VIRUS-KILLER V1.0; by SM'87¶
¶
start:¶
 move.l 4,a6     ;EXECBASE dans A6¶
 moveq #0,d1     ;Flags: Pas de virus à cet endroit¶
 tst.l 46(a6)    ;Test sur altération de Cool-Capture¶
 beq.s noSCA     ;Ce n'était pas un virus SCA¶
 clr.l 46(a6)    ;Initialisation de Cool-Capture¶
 addq.b #1,d1    ;Bit 0 positionné¶
noSCA:¶
```

```
   cmpi.w #$FC,148(a6) ;Interruption de temps mort
vertical¶
   beq.s noVBI       ;non?¶
   addq.b #2,d1      ;Bit 1 positionné¶
   bra.s ClearTag    ;Initialisation de KickTag-
Pointer¶
noVBI:¶
   tst.l 550(a6)     ;KickTag-Pointer modifié?¶
   beq.s GoOn        ;non?¶
   addq.b #4,d1      ;Bit 2 positionné¶
ClearTag:¶
   clr.l 550(a6)     ;Pointeur initialisé¶
   clr.l 554(a6)     ;Pointeur initialisé¶
GoOn:¶
   move.b d1,Virusflag ;Drapeau réservé¶
   lea dosname,a1      ;Adresse des noms de
bibliothèques¶
   moveq #0,d0         ;La version est la même¶
   jsr -552(a6)        ;OpenLibrary¶
   move.l d0,dosbase   ;Réserve Library-Base¶
   beq errfix          ;Branchement sur erreur¶
   move.l d0,a6        ;Preparation pour appel au
DOS¶
   jsr -60(a6)         ;Lecture Output-handle¶
   move.l d0,Outputhandle ;et enregistrement¶
   beq errfix          ;Branchement si erreur¶
   move.l #title,d2    ;Titre dans d2¶
   move.l #titleend-title,d3 ;Long. du texte¶
   jsr writeout        ;Sortie du texte¶
   tst.b Virusflag     ;Test du drapeau de virus¶
   bne.s Virusfound    ;Branchement sur virus¶
   move.l #clean,d2    ;Suppression du message¶
   move.l #cleanend-clean,d3 ;Longueur¶
   jsr writeout        ;Sortie du texte¶
   bra errfix          ;Fin du programme¶
Virusfound:          ;Virus est actif¶
   btst #0,Virusflag   ;Cool-Capture?¶
   beq.s notsca        ;Non¶
   move.l #scaV,d2     ;Message¶
   move.l #scaVend-scaV,d3 ;Longueur¶
   jsr writeout        ;Sortie du texte¶
notsca:¶
   btst #1,Virusflag   ;Interruption de temps mort
vertical?¶
   beq.s notvbi        ;Non¶
   move.l #bbVvbi,d2   ;Message¶
   move.l #bbVvbiend-bbVvbi,d3 ;Longueur¶
   jsr writeout        ;sortie¶
   bra.s bbfound       ;Message suivant¶
notvbi:¶
   btst #2,Virusflag   ;Kicktag?¶
   beq.s errfix        ;Non¶
   move.l #bbVtag,d2   ;Message¶
```

```
   move.l #bbVtagend-bbVtag,d3 ;Longueur¶
   jsr writeout          ;sortie¶
bbfound:¶
   move.l #bbV,d2        ;message¶
   move.l #bbVend-bbV,d3 ;Longueur¶
   jsr writeout          ;sortie¶
errfix:¶
   move.l 4,a6¶
   move.l dosbase,d0¶
   beq.s quit¶
   move.l d0,a1¶
   jsr -414(a6)¶
quit:¶
   moveq #0,d0¶
   rts¶
writeout:¶
   move.l Outputhandle,d1¶
   jmp -48(a6)¶
dosname:dc.b "dos.library",0¶
title:dc.b $c,$9b,"1;31;42m - Virus-Killer"¶
      dc.b " V1.0 - ",10,13¶
      dc.b "(c)  1988 par S. Maelger",10,13,10¶
      dc.b $9b,"0;31;40m"¶
titleend:even¶
clean:dc.b "Pas de symptôme de virus"¶
      dc.b "-Infection trouvée!",10,13,10¶
cleanend:even¶
scaV:dc.b "Reset-Vector Cool-Capture a"¶
     dc.b " été utilisé!",10,13¶
     dc.b "Virus SCA suspecté !",10,13¶
     dc.b "Virus en mémoire détruit.",10,13,10¶
scaVend:even¶
bbVvbi:dc.b "Vertical Blank Interrupt a "¶
       dc.b "été utilisé!",10,13¶
bbVvbiend:even¶
bbV:dc.b "Virus Byte-Bandit suspecté!",10,13¶
    dc.b "Virus en mémoire détruit.",10,13,10¶
bbVend:even¶
bbVtag:dc.b "KickTagPointer n'est plus"¶
       dc.b " dans le système d'exploitation!",10,13¶
bbVtagend:even¶
dosbase:dc.l 0¶
Outputhandle:dc.l 0¶
Virusflag:dc.b 0¶
   end¶
```

Voici la version du programme de chargement BASIC du programme de détection des virus présenté ci dessus.

```
OPEN "sys:c/Virus_chk" FOR OUTPUT AS 1¶
FOR i=1 TO  812¶
  READ a$¶
  a%=VAL("&H"+a$)¶
  PRINT #1,CHR$(a%);¶
NEXT¶
CLOSE 1¶
KILL "sys:c/Virus_chk.info"¶
datas:¶
DATA 00,00,03,F3,00,00,00,00,00,00,00,01,00,00,00,00¶
DATA 00,00,00,00,00,00,00,A8,00,00,03,E9,00,00,00,A8¶
DATA 2C,79,00,00,00,04,72,00,4A,AE,00,2E,67,06,42,AE¶
DATA 00,2E,52,01,0C,6E,00,FC,00,94,67,04,54,01,60,08¶
DATA 4A,AE,02,26,67,0A,58,01,42,AE,02,26,42,AE,02,2A¶
DATA 13,C1,00,00,02,9C,43,F9,00,00,01,16,70,00,4E,AE¶
DATA FD,D8,23,C0,00,00,02,94,67,00,00,AA,2C,40,4E,AE¶
DATA FF,C4,23,C0,00,00,02,98,67,00,00,9A,24,3C,00,00¶
DATA 01,22,26,3C,00,00,00,48,4E,B9,00,00,01,0C,4A,39¶
DATA 00,00,02,9C,66,16,24,3C,00,00,01,6A,26,3C,00,00¶
DATA 00,2E,4E,B9,00,00,01,0C,60,00,00,6A,08,39,00,00¶
DATA 00,00,02,9C,67,12,24,3C,00,00,01,98,26,3C,00,00¶
DATA 00,5C,4E,B9,00,00,01,0C,08,39,00,01,00,00,02,9C¶
DATA 67,14,24,3C,00,00,01,F4,26,3C,00,00,00,2A,4E,B9¶
DATA 00,00,01,0C,60,1C,08,39,00,02,00,00,02,9C,67,24¶
DATA 24,3C,00,00,02,58,26,3C,00,00,00,3C,4E,B9,00,00¶
DATA 01,0C,24,3C,00,00,02,1E,26,3C,00,00,00,3A,4E,B9¶
DATA 00,00,01,0C,2C,79,00,00,00,04,20,39,00,00,02,94¶
DATA 67,06,22,40,4E,AE,FE,62,70,00,4E,75,22,39,00,00¶
DATA 02,98,4E,EE,FF,D0,64,6F,73,2E,6C,69,62,72,61,72¶
DATA 79,00,0C,9B,31,3B,33,31,3B,34,32,6D,20,2D,20,56¶
DATA 69,72,75,73,2D,4B,69,6C,6C,65,72,20,56,31,2E,30¶
DATA 20,2D,20,0A,0D,28,63,29,20,20,31,39,38,38,20,70¶
DATA 61,72,20,53,2E,20,4D,61,65,6C,67,65,72,0A,0D,0A¶
DATA 9B,30,3B,33,31,3B,34,30,6D,00,50,61,73,20,64,65¶
DATA 20,73,79,6D,70,74,F4,6D,65,20,64,65,20,76,69,72¶
DATA 75,73,2D,49,6E,66,65,63,74,69,6F,6E,20,74,72,6F¶
DATA 75,76,E9,65,21,0A,0D,0A,52,65,73,65,74,2D,56,65¶
DATA 63,74,6F,72,20,43,6F,6F,6C,2D,43,61,70,74,75,72¶
DATA 65,20,61,20,E9,74,E9,20,75,74,69,6C,69,73,E9,21¶
DATA 0A,0D,56,69,72,75,73,20,53,43,41,20,73,75,73,70¶
DATA 65,63,74,E9,20,21,0A,0D,56,69,72,75,73,20,65,6E¶
DATA 20,6D,E9,6D,6F,69,72,65,20,64,E9,74,72,75,69,74¶
DATA 2E,0A,0D,0A,56,65,72,74,69,63,61,6C,20,42,6C,61¶
DATA 6E,6B,20,49,6E,74,65,72,72,75,70,74,20,61,20,E9¶
DATA 74,E9,20,75,74,69,6C,69,73,E9,21,0A,0D,00,56,69¶
DATA 72,75,73,20,42,79,74,65,2D,42,61,6E,64,69,74,20¶
DATA 73,75,73,70,65,63,74,E9,21,0A,0D,56,69,72,75,73¶
DATA 20,65,6E,20,6D,E9,6D,6F,69,72,65,20,64,E9,74,72¶
DATA 75,69,74,2E,0A,0D,0A,00,4B,69,63,6B,54,61,67,50¶
```

```
DATA 6F,69,6E,74,65,72,20,6E,27,65,73,74,20,70,6C,75¶
DATA 73,20,64,61,6E,73,20,6C,65,20,73,79,73,74,E8,6D¶
DATA 65,20,64,27,65,78,70,6C,6F,69,74,61,74,69,6F,6E¶
DATA 21,0A,0D,00,00,00,00,00,00,00,00,00,00,00,00,00¶
DATA 00,00,03,EC,00,00,00,16,00,00,00,00,00,00,00,32¶
DATA 00,00,00,38,00,00,00,44,00,00,00,54,00,00,00,5E¶
DATA 00,00,00,6A,00,00,00,70,00,00,00,78,00,00,00,84¶
DATA 00,00,00,90,00,00,00,98,00,00,00,A4,00,00,00,AC¶
DATA 00,00,00,B4,00,00,00,C0,00,00,00,CA,00,00,00,D2¶
DATA 00,00,00,DE,00,00,00,E4,00,00,00,F0,00,00,00,FC¶
DATA 00,00,01,0E,00,00,00,00,00,00,03,F2¶
```

3.3. Le langage machine et BASIC

Pour appeler les routines du langage machine sous BASIC, une variable longue est utilisée pour transférer l'adresse de départ de la routine. Nous allons en faire la démonstration sur la routine Reset du système d'exploitation qui débute à l'adresse $FC0000. Malheureusement, BASIC pose des problème lors du traitement des variables longues. L'interpréteur BASIC ne calcule presque exclusivement que des variables en virgule flottante qu'il convertit ensuite en variables longues.

L'erreur fréquemment rencontrée provient du fait que les variables en virgule flottante ne sont précises que sur quelques décimales seulement. Par conséquent, le résultat de la conversion est inférieur d'une valeur comprise entre 1 et 5. Pour contourner ce problème, vous devez soit employer une arithmétique de valeurs longues plus précises, soit utiliser des chaînes:

```
SMreset&=CVL(CHR$(0)+CHR$(&HFC)+MKI$(0)¶
```

Lors d'une utilisation fréquente de routines système et de programmes en langage machine, vous avez la possibilité de déclarer toutes les variables dont l'étiquette est plus courte:

```
DEFLNG a-z¶
SMreset=CVL(CHR$(0)+CHR$(&HFC)+MI(0))¶
SMreset¶
```

Remarque : Attention ! Lorsque vous entrez l'exemple ci-dessous et que vous l'exécutez, l'interpréteur BASIC se branche sur la routine Reset. Ceci constitue uniquement un

exemple de branchement sur une routine du langage machine; cette routine a pour effet de réinitialiser l'ordinateur. Si vous voulez effectuer une réinitialisation dans un programme BASIC, il existe une méthode plus efficace et plus rapide:

```
POKEL 32,CVL(CHR$(0)+CHR$(&HFC)+MKI$(0)¶
```

Mais, nous nous éloignons du sujet. Revenons au langage machine. Pour aller plus loin dans le perfectionnement des commandes BASIC, nous avons écrit une courte routine qui allume et éteint la diode électroluminescente de Mise sous Tension. C'est ce que fait l'Amiga quand vous le réinitialisez (ou lorsqu'il se réinitialise de lui même). Pour l'essentiel, la routine est la suivante:

```
Code                          Mnémonique¶
089000100BFE001 BCHG # 1,   $BFE001 ; allumer diode¶
4E5             RTS                 ; c'est tout¶
```

Il s'agit à présent de savoir où placer le programme. La réponse n'est pas aussi simple qu'il parait car l'adresse de départ du code doit toujours être paire. Dans le cas contraire, un Guru#3 est généré (erreur d'adressage). Le processeur 68000 ne traite que les commandes situées à des adresses paires. Ici, BASIC ne donne aucune indication sur ce qui se produit et place ses variables ,au niveau de l'octet, dans la mémoire tampon appropriée s'il y a suffisamment de place en mémoire. Pour être absolument certain que l'adresse est paire, l'emplacement en mémoire doit être attribué par le système. La routine AllocMem d'Exec permet de parvenir à ce résultat:

```
' LED-FLICKER.BAS¶
DEFLNG a-z¶
DECLARE FUNCTION AllocMem LIBRARY¶
LIBRARY"t&a2:bmaps/exec.library"¶
SMmagic=AllocMem(10,1) '10 octets, memory area public
RAM¶
FOR i%=0 TO 4¶
 READ Power$¶
 POKEW SMmagic+i%*2,VAL("&H"+Power$)¶
NEXT¶
DATA 879, 1, BF, E001, 4E75¶
PRINT "Observez le clignotement de la diode de mise
sous tension"¶
FOR i%=1 TO 20 '20 clignotements¶
 a!=TIMER+.5 'toutes les 0,5 secondes¶
 WHILE a!TIMER:WEND 'attendez¶
```

```
 SMmagic 'appelez la routine assembleur¶
NEXT¶
FreeMem SMmagic,10 'emplacement mémoire libre¶
LIBRARY CLOSE 'fermez SYS¶
```

La routine en elle-même ne pose pas de problème ; elle n'utilise aucun de ses propres registres. Tous les assembleurs que l'on trouve sur le marché acceptent l'exemple suivant :

```
MOVE.L D0,Dataregister0¶
...¶
Dataregister0:dc.l 0¶
```

Cette routine définit une étiquette au sein du programme, dans laquelle le registre de données D0 doit être placé. Lorsque les codes correspondants, similaires à ceux du programme de la diode de Mise sous Tension, sont écrits en mémoire par un poke, vous pouvez vous attendre à une Méditation du Gourou dans les plus brefs délais. La méthode d'adressage employée dans MOVE.L est à l'origine de l'erreur.

La méthode d'adressage absolu retenue est inadéquate car on ne sait jamais à quel endroit de la mémoire le programme va échouer une fois chargé.

Le code doit d'abord s'approcher de l'adresse de départ que l'on veut utiliser.Un bon assembleur doit permettre de programmer en se servant du contenu du compteur ordinal comme adresse de référence. Dans tous les cas, vous devez écrire:

```
LEA Dataregister0(PC),A0¶
MOVE.L DO, (A0)¶
```

Si vous programmez uniquement en employant cette méthode d'adressage, votre programme ne pourra pas utiliser toutes les instructions de l'assembleur et il sera trés long. Nous allons vous montrer comment utiliser toutes les instructions du langage machine en évitant totalement les affectations de mémoire, en contournant les routines de chargement sophistiquées et en réalisant des appels fonction aux programmes C.

3.3.1. Assembleur et programmes C à partir de BASIC

Nous ne nous bornerons pas ici à vous donner les rudiments de ce type d'accès.Bien au contraire, nous allons étudier le programme (non BASIC) dans son intégralité. Qu'il s'agisse d'un jeu, d'un traitement de texte ou d'un interpréteur BASIC, le mode d'adressage employé pour les programmes est l'adressage absolu bien que le 68000 exige un adressage relatif pour placer les programmes à un endroit quelconque de la mémoire. Si vous pensez à cela pendant un instant, vous parviendrez à la conclusion que les adresses d'un programme se présentent uniquement sous la forme de déplacements. Peu après avoir chargé le programme, le système d'exploitation calcule les adresses correctes de façon à ce que le programme puisse s'ajuster aux adresses dans lesquelles il sera chargé.

Le problème est de savoir comment anticiper cette action. Le système d'exploitation sait quelles commandes doivent être codées. Ces informations sont sauvegardées au moyen de l'éditeur de liens qui les tient de l'assembleur. Généralement, un programme assembleur sauvegardé consiste en un segment de code (votre programme), un segment de données et un segment BSS (qui réserve les emplacement mémoire disponibles pour le programme directement après son chargement), ainsi que un ou plusieurs segments de chaînage.

Vous pouvez examiner les opérations réalisées par le système d'exploitation en utilisant la routine de chargement pour démarrer le programme (votre programme ne doit pas démarrer automatiquement). Cette routine figure dans dos.library et peut être appelée à partir de BASIC sans difficulté. La routine porte le nom de LoadSeg et exige que le nom du programme se termine par un octet nul.LoadSeg se charge de l'affectation de la mémoire et du reste. La syntaxe est la suivante:

```
MainSegment=LoadSeg ( SADD( "Nom du fichier" +
CHR$(0)))¶
```

Pour libérer les emplacements mémoire réservés, il faut invoquer la routine UnloadSeg et lui donner la valeur de retour de LoadSeg.

```
UnloadSeg MainSegment¶
```

La valeur renvoyée par LoadSeg pose quelques problèmes. AmigaDOS étant écrit en BCPL (langage de compilation), cette valeur est traitée comme une valeur BCPL. Le langage BCPL se réfère à cette valeur comme étant un segment Main ou Code. Vous devez convertir cette valeur en une adresse pour vos besoins propres. DOS se comporte ,à cet égard, comme si la mémoire était formée uniquement de valeurs longues placées les unes à la suite des autres. Par conséquent, il traite le numéro de la valeur longue à partir du début de la mémoire sous la forme de la valeur de retour.

Pour obtenir une adresse correcte, cette valeur doit être multipliée par quatre (quatre octets par valeur longue). Cette adresse n'est pas exactement donnée dans le programme. Le pointeur BCPL sur le segment suivant est enregistré dans la première valeur longue du segment principal. Ce segment débute par un autre pointeur BCPL sur un autre segment, et ainsi de suite. Le programme apparaît à la suite de ce pointeur. Utilisez , par conséquent, la routine suivante pour accéder à une routine en langage machine :

```
DEFLNG a-z¶
DECLARE FUNCTION loadseg LIBRARY¶
LIBRARY"dos.library"¶
'changez le nom du fichier de votre programme en
langage machine¶
bcpl.segstart=loadseg(SADD("df1:asm.prg"+
CHR$(0)))¶
Segment=bcpl.segstart*4¶
Routine=Segment+4¶
Routine¶
unloadseg bcpl.segstart¶
LIBRARY CLOSE¶
```

Pour visualiser les adresses de départ des segments individuels, insérer les lignes suivantes avant l'appel de Routine :

```
PRINT "Le programme principal débute à l'adresse";
Routine¶
i%=1¶
WHILE Segment<>0¶
 PRINT i%;". Le segment débute à l'adresse";Segment+
4¶
 Segment=PEEKL(Segment)*4¶
WEND¶
```

Nous allons parler maintenant du passage des paramètres. BASIC traite les routines assembleur de la même manière que le langage C traite les segments de programme. Les paramètres sont enregistrés sous la forme de valeurs longues sur la pile. Dans l'exemple suivant, le paramètre P3 est introduit sur la pile en premier, puis P2 et enfin P1. Le paramètre P1 sera donc le premier sorti de la pile dans le programme appelé. A la suite des paramètres, le branchement vers sous-programme (au moyen de JSR-Jump to SubRoutine-) renvoie le code à la dernière adresse de branchement de retour dans la pile :

```
Routine P1,P2,P3¶
```

L'adresse de branchement de retour de la routine est sur la pile :

```
Pointeur de pile + 12 = P3 (Long)¶
Pointeur de pile + 8  = P2 (Long)¶
Pointeur de pile + 4  = P1 (LOng)¶
Pointeur de pile + 0  = adresse de branchement de
retour (au moyen de JSR).¶
```

Lorsque vous chargez une routine en C avec le programme ci-dessus, vous pouvez l'appeler exactement de la même manière que vous le feriez avec un programme assembleur. Chaque fois que possible, il est préférable d'utiliser le langage machine car un code machine généré par C est plus lent que le langage machine direct. Certains compilateurs BASIC créent des codes exécutés plus rapidement qu'en C car les routines du système d'exploitation sont appelées exactement de la même façon que les commandes BASIC ont été écrites.

Certaines parties du système d'exploitation sont programmées en C, c'est ce qui explique la lenteur de l'Amiga. C utilise beaucoup d'espace mémoire. Seule une petite partie du système d'exploitation est écrite en C. Cette partie comprend la plus grande portion de mémoire nécessaire au système d'exploitation.
C'est pourquoi , la taille du système d'exploitation diminue en même temps qu'il devient plus rapide. Dès que les concepteurs de l'Amiga ont l'occasion d'apporter des retouches au système, ils remplacent les routines en C par des routines en langage machine.

En C, les paramètres sont inscrits entre parenthèses dans le même ordre que sous BASIC. Comment passe-t-on du langage machine à C? Il faut commencer par indiquer que vous ne pouvez modifier aucun des registres sans avoir au préalable sauvegarder leurs valeurs courantes. Ces valeurs doivent être enregistrées au début du programme:

```
START: MOVEMEM.L D0-A6,-(A7)   ; sauvegardez les¶
                               ; registres:¶
                               ; Vous trouverez 15¶
                               ; registres¶
                               ; et l'adresse de¶
                               ; branchement¶
                               ; de retour sur la¶
                               ; pile.¶
                               ; le premier paramètre¶
                               ; se trouve à¶
                               ; SP+(16*4) = 645SP)¶
MOVEM.L 64(SP),D0-D2           ; prendre les 3¶
                               ; paramètres¶
¶
...¶
END: MOVEM.L (A7)+,D0-A6       ; Registre pris dans
la¶
                               ; pile¶
RTS¶
```

Vous vous demandez surement par quel biais on accède aux valeurs renvoyées au programme BASIC. On passe les adresses d'une variable sous la forme d'un paramètre et on accède aux valeurs renvoyées par l'intermédaire des fonctions VARPTR ou Sadd. L'adresse est extraite de la pile dans le programme en langage machine et écrite dans la valeur de retour. Lorsque plusieurs valeurs sont renvoyées, une adresse est attribuée à une variable de type tableau (il s'agit d'être prudent avec les chaînes; ici l'adresse obtenue est celle du descripteur de chaîne, qui est formée de cinq octets).

3.3.2. Perfectionnement de BASIC : ColorCycle

Nous avons choisi comme illustration du passage des paramètres dans un programme en langage machine , l'amélioration suivante apportée à AmigaBASIC. Ce programme permet la rotation des couleurs comme dans DPaint.

```
  ;---------------------------------------- ;¶
  ; BASIC-Extension   ColorCycle  SMmagic'88  ;¶
  ;----------------------------------------;¶
  ; Syntaxe: AddressRoutine WINDOW(7),de ,à   ;¶
  ; de =1ere couleur; à =dernière couleur     ;¶
  ; Toujours  "de" --> "à"                    ;¶
  ; de < à : Rotation vers le haut            ;¶
  ; de > à : Rotation vers le bas             ;¶
  ;----------------------------------------;¶
Cycle:    MOVEM.L D0-A6,-(SP);Réserve registre sur pile¶
          MOVE.L 4,A6        ;Lit ExecBase dans A6¶
          LEA GFXNAME,A1     ;Nom des Biblio. dans A1¶
          MOVEQ #0,D0        ;Version est la même¶
          JSR -552(A6)       ;Appel à OpenLibrary¶
          TST.L D0           ;Test, Base est-il
disponible¶
          BEQ.S Exit         ;Si non, alors fin¶
          MOVE.L D0,A6       ;GfxBase dans A6¶
          MOVE.L 64(SP),A0   ;Prise en charge de
WindowBase¶
          MOVE.L 46(A0),A0   ;ScreenBase determinée¶
          ADD.L #44,A0       ;ViewPort de Screens
dans
A0¶
          MOVE.L 4(A0),A1    ;ColorTable dans A1¶
          MOVE.L 4(A1),A1    ;ColorMap determiné¶
          LEA CTab,A2        ;Buffer utilisateur dans
A2¶
          MOVEQ #15,D0       ;15 Longs (32 Mots)¶
CopyCT:  MOVE.L (A1)+,(A2)+;ColorMap copié¶
         DBRA D0,CopyCT     ;(si non, alors¶
          MOVEM.L 68(SP),D0-D1 ;lire couleur de début
et de fin¶
          ANDI.W #31,D0      ;si la valeur dépasse 31¶
          ANDI.W #31,D1      ;idem¶
          LSL.B #1,D0        ;*2 (utilise comme décalage)¶
          LSL.B #1,D1        ;idem¶
          LEA CTab,A1        ;Adresse de notre mém.
tampon¶
          MOVE.W (A1,D1.W),D2 ;réserve dernière
couleur¶
          CMP.B D0,D1        ;détermine sens de
rotation¶
          BEQ.S ClLib        ;les deux couleurs sont
les mêmes?¶
          BGT.S Up           ;Rotation des coul. vers
le haut¶
Down:     MOVE.W 2(A1,D1.W),(A1,D1.W);puis vers le
bas¶
         ADDQ.B #2,D1          ;incrément du décalage¶
          CMP.B D0,D1          ;C'est la fin?¶
          BNE.S Down           ;non? alors couleur
suivante¶
```

```
          BRA.S SetLC          ;fermé¶
Up:       MOVE.W -2(A1,D1.W),(A1,D1.W) ;couleurs vers
le bas¶
          SUBQ.B #2,D1         ;incrément du décalage¶
           CMP.B D0,D1          ;c'est la fin¶
           BNE.S Up             ;non? alors couleur
suivante¶
SetLC:    MOVE.W D2,(A1,D1.W) ;couleur réservée¶
          MOVEQ #32,D0         ;32 couleurs¶
          JSR -192(A6)         ;LoadRGB4 (a0=VP,a1=
Ctab,d0)¶
ClLib:    MOVE.L A6,A1         ;GfxBase dans A1¶
¶
          MOVE.L 4,A6          ;Lire ExecBase¶
          JSR -414(A6)         ;Bibliothèque fermée¶
Exit:     MOVEM.L (SP)+,D0-A6 ;Registre à partir de la
pile¶
          RTS                 ;Fin¶
GFXNAME: DC.B "graphics.library",0,0 ;Library-Name¶
CTab:     DS.W 32             ;Mémoire tampon de 32 mots¶
          END¶
```

Le code est configuré de telle façon que vous puissiez l'assembler en mode normal ou en prenant comme référence les adresses du compteur ordinal. Le programme BASIC ci-dessous appelle la routine ColorCycle :

```
'Chargez d'abord la routine:¶
DEFLNG a-z¶
DECLARE FUNCTION loadseg LIBRARY¶
LIBRARY"T&A2:bmaps/dos.library"¶
a=loadseg(SADD("T&A2:ColorCycle"+CHR$(0)))¶
prg=a*4+4¶
'Un petit graphique¶
FOR i%=0 TO 3¶
 LINE (0,i%*40)-STEP(80,40),i%,bf¶
NEXT¶
'Demo: Rotation des couleurs vers l'avant et vers
l'arrière¶
FOR i%=0 TO 50¶
 t!=TIMER+.2¶
 WHILE t!>TIMER¶
 WEND¶
 prg WINDOW(7),1,3¶
NEXT¶
FOR i%=0 TO 50¶
 t!=TIMER+.2¶
 WHILE t!>TIMER¶
 WEND¶
 prg WINDOW(7),3,1¶
NEXT¶
'Libération de la mémoire¶
```

```
unloadseg a¶
LIBRARY CLOSE¶
¶
Le programme de chargement BASIC suivant génère le
fichier ColorCycle sur le disque:¶
¶
¶
OPEN "COLORCYCLE" FOR OUTPUT AS 1¶
FOR i=1 TO  300¶
 READ a$¶
 a$="&H"+a$¶
 PRINT#1,CHR$(VAL(a$));¶
NEXT¶
CLOSE 1¶
DATA 0,0,3,F3,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,40,0,0,3A,0,0¶
DATA 3,E9,0,0,0,3A,48,E7,FF,FE,2C,78,0,4,43,F9,0,0,0,94,70,0¶
DATA 4E,AE,FD,D8,4A,80,67,76,2C,40,20,6F,0,40,20,68,0,2E,D1¶
DATA FC,0,0,0,2C,22,68,0,4,22,69,0,4,45,F9,0,0,0,A6,70,F,24¶
DATA D9,51,C8,FF,FC,4C,EF,0,3,0,44,2,40,0,1F,2,41,0,1F,E3,8¶
DATA E3,9,43,F9,0,0,0,A6,34,31,10,0,B2,0,67,26,6E,E,33,B1,10¶
DATA 2,10,0,54,1,B2,0,66,F4,60,C,33,B1,10,FE,10,0,55,1,B2,0¶
DATA 66,F4,33,82,10,0,70,20,4E,AE,FF,40,22,4E,2C,78,0,4,4E¶
DATA AE,FE,62,4C,DF,7F,FF,4E,75,67,72,61,70,68,69,63,73,2E¶
DATA 6C,69,62,72,61,72,79,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0¶
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0¶
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,EC¶
DATA 0,0,0,3,0,0,0,0,0,0,0,A,0,0,0,32,0,0,0,52,0,0,0,0,0,0,3¶
DATA F2,0,0,3,F2¶
DATA BECKER¶
```

3.3.3. Inactiver la souris

Lorsque Amiga exécute une opération sur le disque, un pictogramme d'attente apparaît (un nuage avec des Zs dessinés à l'intérieur). Les programmes professionnels ont des pointeurs de souris de forme différente. Notre objectif est d'inactiver la souris en BASIC. Ceci est utile lors des opérations sur disque ou de la lecture de données. Voici le programme:

```
  ;-------------------------------------------------;¶
  ;BASIC-Extension Zzz                    SMmagic'88;¶
  ;-------------------------------------------------;¶
  ;Syntax: Zzz WINDOW(7),OnOff                      ;¶
  ;OnOff  paire ou 0  :    La souris est désactivée;¶
  ;OnOff  odd     (1) :    La souris est normale   ;¶
  ;-------------------------------------------------;¶
ZZZ:      MOVEM.L D0-A6,-(SP);Réserve registre¶
          MOVE.L 4,A6         ;Lit l'adresse de base de
```

```
Exec¶
        LEA INTNAME,A1  ;Adresse des noms de
biblio.¶
        MOVEQ #0,D0     ;La version est la même¶
        JSR -552(A6)    ;OpenLibrary¶
        TST.L D0        ;Non, que se passe-t-il?¶
        BEQ.S ENDE      ;Non? alors fin¶
        MOVE.L D0,A6    ;IntuitionBase chargé¶
        MOVE.L 64(SP),A0    ;Lecture de  WindowBase¶
        MOVE.L 68(SP),D0    ;Lecture OnOff-Flag¶
        BTST #0,D0      ;Test du bit 0¶
        BEQ.S SLEEP     ;Nombre paire? Bonne nuit¶
        JSR -60(A6)     ;ClearPointer appelé¶
        BRA.S EXIT      ;et fin¶
SLEEP:  LEA MOUSE,A1;Adresse des données du pointeur¶
        MOVEQ #22,D0        ;Hauteur en d0¶
        MOVEQ #16,D1        ;Largeur en d1¶
        MOVEQ #0,D2         ;Suppression du
décalage horiz.¶
        MOVE.L D2,D3        ;Suppression du
décalage vert.¶
        JSR -270(A6)        ;SetPointer appellé¶
EXIT:   MOVE.L A6,A1        ;IntuitionBase dans a1¶
        MOVE.L 4,A6         ;ExecBase¶
        JSR -414(A6)        ;Appel de CloseLibrary¶
ENDE:   MOVEM.L (SP)+,D0-A6 ;Registre de la pile¶
        RTS                 ;Retour au programme BASIC¶
INTNAME: DC.B "intuition.library",0 ;Library-Name¶
MOUSE:  DC.L 0,$3000300,$7A007A0,$1FF01FF0,$3FF03FF0¶
        DC.L $30F83FF8,$3DFC3FFC,$7BFC7FFC,$30FE3FFE¶
        DC.L $3F863FFE,$1FEF1FFF,$3FDE3FFE,$1F861FFE¶
        DC.L $FFC0FFC,$3F803F8,$E000E0,$3800380,
$7E007E0¶
        DC.L $3400340,0,$600060,$700070,$200020,0¶
        END                 ;Fin des données¶
;Appel typique sous BASICBASIC:¶
;ZZZ WINDOW(7),0     'inactivée¶
;ZZZ WINDOW(7),1     'normale¶
```

Le générateur BASIC suivant crée le code machine de cette routine :

```
OPEN "ZZZ" FOR OUTPUT AS 1¶
FOR i=1 TO  260¶
 READ a$¶
 a$="&H"+a$¶
 PRINT#1,CHR$(VAL(a$));¶
NEXT¶
CLOSE 1¶
DATA 0,0,3,F3,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,40,0,0,31,0,0,3¶
DATA E9,0,0,0,31,48,E7,FF,FE,2C,78,0,4,43,F9,0,0,0,50,70,0,4E¶
```

```
DATA AE,FD,D8,4A,80,67,32,2C,40,20,6F,0,40,20,2F,0,44,8,0,0,0¶
DATA 67,6,4E,AE,FF,C4,60,12,43,F9,0,0,0,62,70,16,72,10,74,0¶
DATA 26,2,4E,AE,FE,F2,22,4E,2C,78,0,4,4E,AE,FE,62,4C,DF,7F,FF¶
DATA 4E,75,69,6E,74,75,69,74,69,6F,6E,2E,6C,69,62,72,61,72,79¶
DATA 0,0,0,0,0,3,0,3,0,7,A0,7,A0,1F,F0,1F,F0,3F,F0,3F,F0,30¶
DATA F8,3F,F8,3D,FC,3F,FC,7B,FC,7F,FC,30,FE,3F,FE,3F,86,3F,FE¶
DATA 1F,EF,1F,FF,3F,DE,3F,FE,1F,86,1F,FE,F,FC,F,FC,3,F8,3,F8¶
DATA 0,E0,0,E0,3,80,3,80,7,E0,7,E0,3,40,3,40,0,0,0,0,0,60,0¶
DATA 60,0,70,0,70,0,20,0,20,0,0,0,0,0,0,0,3,EC,0,0,0,2,0,0¶
DATA 0,0,0,0,0,A,0,0,0,30,0,0,0,0,0,0,3,F2,0,0,3,F2¶
DATA BECKER¶
```

Le fichier est chargé au moyen de la routine LoadSegment. Voici un programme de démonstration en BASIC qui charge la routine :

```
'Chargez la routine d'abord:¶
DEFLNG a-z¶
DECLARE FUNCTION loadseg LIBRARY¶
LIBRARY"T&A2:bmaps/dos.library"¶
a=loadseg(SADD("T&A2:Progs/ZZZ"+CHR$(0)))¶
prg=a*4+4¶
'Demo: La souris est au repos¶
¶
 prg WINDOW(7),0  :'La souris est au repos¶
¶
FOR i%=0 TO 5000¶
NEXT¶
¶
prg WINDOW(7),1 :'La souris est normale¶
¶
'Libération de la mémoire¶
unloadseg a¶
LIBRARY CLOSE¶
```

Chapitre 4

Incursion dans le matériel

Pourquoi l'Amiga absorbe-t-il tout votre temps ? Ses composants logiciels et matériels en font un ordinateur de qualité. Le présent chapitre est consacré à la partie matérielle ou hardware de l'Amiga. Vous y apprendrez toutes sortes de choses ingénieuses et de techniques destinées à améliorer l'aspect matériel de votre ordinateur.

Avant d'aller plus loin, quelques mises au point s'imposent :

❶ L'objectif visé par le présent chapitre n'est en aucun cas de vous enseigner l'électronique. Nous supposons que vous possédez certaines connaissances en électronique, circuits et composants et que vous savez manipuler un fer à souder et un tournevis. Si ce n'est pas le cas, il est préférable de combler ces lacunes avant de commencer à démonter votre Amiga. Si malgré tout, vous n'êtes pas très sûr de vous, NE TOUCHEZ A RIEN ! Confiez plutôt cette mission à une personne compétente. Un raccord mal fait risque d'endommager votre ordinateur.

❷ L'ouverture d'un châssis provoque la nullité de la garantie (si elle est toujours en vigueur). Toute intervention sur le matériel de la part d'un utilisateur va à l'encontre de ce qui est stipulé dans le contrat de garantie. Cela signifie que le fabricant ou le revendeur ne sont plus tenus de réparer la machine à leurs frais. En d'autres termes, si vous mettez votre ordinateur hors d'état, les frais de réparation seront à votre charge, même si la garantie n'est pas encore arrivée à son terme.

❸ Nous avons testé toutes les modifications décrites ici. Il n'est cependant pas toujours possible pour un auteur de vérifier l'intégralité des versions commercialisées d'un ordinateur. Il se peut, par conséquent, qu'une ou deux erreurs se soient glissées dans le chapitre. Si vous rencontrez des problèmes, remettez les choses à leur état initial, bien qu'en théorie tout doive fonctionner correctement.

❹ Soyez toujours très prudent lorsque vous manipulez des composants électroniques. Dans votre interêt et dans celui de l'ordinateur, ne travaillez jamais sur des fils électriques sans avoir au préalable éteint votre machine (pour plus de sécurité, débranchez la). Vous devez aussi débrancher tout ce qui peut être connecté à l'Amiga, notamment le ou les écrans, souris, clavier et aussi les cartes d'extension. Otez les composants, soudez les, puis reconnectez les avec précaution.

❺ Un dernier point important. Vous n'êtes pas obligé de bricoler votre ordinateur. Si les concepteurs de l'Amiga sont amenés à tenter de telles expériences, l'utilisateur que vous êtes n'y est absolument pas contraint. Si ce dernier avertissement ne vous dissuade pas de trafiquer votre matériel, en avant !

Le moment est venu de sonder les entrailles de votre ordinateur L'extension de la mémoire sera notre premier propos. Quelques petites modifications permettent de configurer la mémoire de façon à ce qu'aucun programme ne soit interrompu. Nous parlerons ensuite des lecteurs de disquettes qu'il est quelquefois necéssaire de pouvoir mettre hors tension immédiatement. Nous en ferons la démonstration. Le fin du fin sera d'apprendre comment équiper votre Amiga d'un processeur 68010. Nous terminerons le chapitre par une discussion sur les autres types de processeurs.

4.1. Mise hors service de l'extension de la mémoire

L'extension de la mémoire offre beaucoup d'avantages. Cependant, son incompatibilité avec de nombreux programmes suscite souvent chez les utilisateurs des mouvements de colère. En réalité, ce problème est lié aux programmes et non pas à la mémoire auxilliaire. Les programmes ne s'exécutent pas pour la bonne raison qu'ils ne savent pas quelle mémoire utiliser. Par exemple, les circuits intégrés pour le son et le graphisme

doivent accéder à la CHIP-RAM. Si un programme les utilise pour accéder à la FAST-RAM, il se "plante". Les tous premiers programmes d'Amiga présentaient ce problème.

Les deux astuces que nous allons vous montrer se proposent de résoudre le problème que l'on vient d'évoquer au niveau matériel. Il existe aussi une solution logicielle (consultez La Bible de l'Amiga, vous y trouverez un programme qui permet de désactiver la FAST-RAM). Il est toutefois plus simple d'allumer l'Amiga, d'actionner un commutateur et de désactiver la FAST-RAM.

4.1.1. La carte 2000A

Si vous possédez un Amiga 2000, la première étape consiste à déterminer s'il est équipé d'une carte 2000A. Dans le cas contraire, ce paragraphe ne vous concerne pas. Identifiez sur la carte, le circuit intégré PAL qui porte l'étiquette U3 (les étiquettes U1 et U6 ne présentent aucun interêt pour nous). Le circuit PAL gère l'organisation de la mémoire disponible. Tout ce que nous avons à faire est d'indiquer au circuit que l'extension de la mémoire doit être désactivée. La commande de libération d'une zone mémoire est transmise par les broches -OVR (position 19) et -SELECT (position 17).

Pour désactiver l'extension de la mémoire, il faut vous assurer que ces deux broches sont déconnectées, en d'autres termes que le courant ne passe pas entre elles. Il existe deux façons de parvenir à cet ojectif:

La première façon qui est aussi la plus simple consiste à briser la connexion entre les broches du circuit PAL et le système. Eteignez l'ordinateur. Déconnectez avec précaution le chemin conducteur de ces broches. Installez un commutateur bipolaire entre les pattes 17 et 19 du circuit et leurs connexions (utilisez des connecteurs sans soudure). Employez un fil suffisamment long et donnez un peu de mou afin de pouvoir monter le commutateur à l'extérieur du châssis. Percez un trou sur le côté de l'ordinateur et placez y le commutateur. Lorsque le commutateur est sur "Marche", le système reconnaît la RAM étendue. Lorsque vous éteignez l'ordinateur pendant 5 à 10 secondes tout en positionnant ce commutateur sur "Arrêt" puis que vous le rallumez, l'Amiga ne reconnaît pas l'extension de la mémoire. Les programmes qui ne tournaient pas avec une mémoire étendue s'exécutent à présent sans causer de problèmes.

ATTENTION : n'actionnez JAMAIS ce commutateur lorsque l'Amiga est sous-tension, et attendez TOUJOURS plusieurs secondes avant de commuter et de rallumer la machine, ceci pour permettre aux condensateurs de se décharger. Les pires plantages systèmes peuvent parfois affecter la machine même après un redémarrage à froid (éteindre et rallumer), certains condensateurs peuvent mettre très longtemps avant de se décharger complètement. Dans ce cas éteignez la machine 30 minutes puis rallumez avant d'appeler au secours. Si au bout des 30 minutes rien ne s'est arrangé vous pouvez appeler au secours.

La seconde méthode est plus sophistiquée mais elle est aussi plus onéreuse et plus difficile à mettre en oeuvre. Le matériel et les outils suivants sont nécessaires:

Matériel :

- Un commutateur bipolaire à une direction
- Environ 30 à 40 cm de double fil torsadé
- Une base (20 broches adaptée au circuit PAL)
- Soudure

Outils :

- Fer à souder
- Couteau pointu ou tournevis
- Pinces

Retirez le circuit intégré avec précaution. Prenez la base et coupez ou brisez net les broches correspondant aux positions 17 et 19. Introduisez la base ainsi modifiée dans l'ancien montage. Reconnectez à présent les broches enlevées au montage à l'aide d'un commutateur bipolaire et de fil. Percez un trou dans le châssis de l'ordinateur de façon à y insérer le commutateur. Assurez vous que le commutateur est solidement connecté. Vérifiez qu'il n'y ait pas de jeu dans les connexions et qu'elles soient bien nettes. Placez ensuite le circuit intégré PAL dans son nouveau montage.

Toutes les connexions doivent être correctes (si elles ne le sont pas, rectifiez les). Testez le fonctionnement du commutateur avant de remonter le châssis.

4.1.2. La carte de circuit imprimé de l'Amiga 500

La conception de la carte de l'Amiga 500 est entièrement différente. Notre objectif est de désactiver la carte 512K de Commodore-Amiga. Cette carte d'extension est munie d'une horloge fonctionnant sur batterie et d'une Fast-RAM. Les opérations que nous allons décrire n'ont aucune incidence sur l'horloge.

Matériel :

- Un commutateur unipolaire à une direction
- Environ 30 à 40 cm de double fil torsadé
- Soudure

Outils :

- Fer à souder
- Couteau pointu ou tournevis

Eteignez l'ordinateur et ouvrez le "tiroir" d'extension. Otez la carte d'extension avec précaution (agissez toujours avec prudence lors du retrait, de la modification ou de l'installation de pièces). Aprés avoir retiré la carte, posez la sur la table devant vous, les sillons vous faisant face (partie de la carte comportant les circuits gravés et les soudures). Vous devez apercevoir les circuits gravés et les soudures et la moitié de la carte à connexions enfichée dans le port d'extension de l'ordinateur.

Localisez la broche 32 de la carte à connexions (attention de ne pas vous tromper). Suivez sa trajectoire jusqu'à la carte de circuit imprimé. Repérez la façon dont la gravure se détache du point de soudure. C'est là notre but. Il faut en quelque sorte briser le chemin conducteur de la gravure. Munissez vous d'un tournevis pointu ou d'un couteau (un couteau de cuisine pointu fera l'affaire). Creusez le sillon pour créer un espace. La coupure doit être nette. Vous devez normalement apercevoir les dessous du sillon dont il ne doit rester aucun résidu à l'endroit de la coupure. L'espace ménagé doit être assez large : environ 3mm. Prenez un morceau de fil double et soudez les deux brins à une extrémité de chaque côté de l'entaille réalisée dans le sillon. Soudez les deux autres extrémités au commutateur unipolaire à une direction. C'est tout.

Réinstallez la carte d'extension avec soin. Assurez vous que toutes les soudures sont "propres" et solides. Connectez l'ensemble et initialisez le Workbench. Lorsque le commutateur est positionné sur "Arrêt", l'Amiga 500 ignore l'extension de la mémoire. Pour la réactiver, il faut éteindre l'Amiga pendant 5 à 10 secondes, positionner le commutateur sur "Marche" et mettre l'ordinateur sous tension. Si l'ordinateur ne réagit pas comme il faut au bon moment, c'est que vous avez commis une erreur. Vérifier les soudures. Si vous n'y prenez pas garde, vos soudures peuvent être froides et les connexions peu solides. Dans ce cas, recommencez les soudures. Aucune autre erreur ne devrait se produire normalement.

4.2. Contrôle des lecteurs de disquette par commutateur

Les lecteurs externes posent autant de problèmes que l'extension de la mémoire. Habituellement, ils sont automatiquement configurés par le système d'exploitation qui utilise la RAM active. Cette autoconfiguration est source de problèmes car AmigaDOS ne peut gérer les données d'un lecteur qu'à l'aide de la Chip-RAM.

Chaque lecteur necessite une mémoire tampon de 30K pour la gestion des fichiers. De nombreux programmes utilisent cette mémoire. Là où le bât blesse, c'est que une fois cette mémoire affectée aux mémoires tampon de disque, les programmes n'y ont plus accès. En conséquence, les programmes s'interrompent en plein milieu ou ils ne peuvent pas du tout être exécutés.

Commutateur "marche/arrêt" pour les lecteurs

La solution envisagée est l'installation d'un commutateur permettant d'activer ou de désactiver un lecteur en fonction des besoins. Munissez vous du matériel suivant:

Matériel :

- Commutateur unipolaire à une direction
- Environ 10 à 20 cm de fil torsadé double
- Soudure

Outils :

- Fer à souder
- Couteau pointu
- Tournevis

Si le lecteur externe n'est pas muni d'un commutateur, il est facile d'en installer un. Le commutateur interrompt la transmission des données en réinitialisant la ligne qui informe l'Amiga qu'un autre lecteur est connecté. La broche 21 (SEL1) sur le connecteur du lecteur gère la sélection du premier lecteur externe.

Eteignez l'Amiga et débranchez le pour plus de sécurité. Localisez le fil correspondant à la broche 21 et coupez le. Soudez les extrémités sectionnées à l'une des extrémités de chaque brin de fil conduisant au commutateur. Soudez les extrémités des fils du commutateur au commutateur unipolaire à une direction soit en réalisant la connexion sur le connecteur lui même soit sur le lecteur. Dans ce dernier cas, percez un trou dans le boîtier du lecteur de façon à y introduire le commutateur. Réassemblez le tout et effectuez un test.

4.3. Installation d'un MC 68010

Nous allons vous donner une chance de vous offrir un Amiga plus rapide pour pas cher. Il suffit tout simplement de remplacer l'ancien 68000 par un 68010, et le tour est joué. Le 68010 de Motorola est compatible à 99,99% avec le 68000. Il ne présente qu'un seul inconvénient qu'il faudra bien accepter.

Passons d'abord en revue les avantages ! Son installation est simple. Pas de soudures (dans la plupart des cas), et pas de composants supplémentaires coûteux. Les tests ont montré que la rapidité de traitement de certaines commandes est accrue de l'ordre de 80%. D'une manière générale, un programme utilise tout autant les commandes rapides que les commandes existantes. L'un dans l'autre, la rapidité de traitement n'est augmentée que de 16% par rapport au processeur 68000, écart qui n'est tout de même pas négligeable.

En outre, vous pourrez rendre votre nouveau processeur compatible à 100% grâce à un programme présenté à la fin de ce chapitre.

Comment installer votre nouveau processeur ? La première des choses à faire est de l'acheter. Son acquisition ne doit pas poser de problème particulier. Il sufit de parcourir les petites annonces des magazines d'informatique ou de la presse spécialisée dans la vente de composants . La dexième solution est de vous adresser directement au magasin d'électronique le plus proche de votre domicile. S'il ne dispose pas de la pièce en question, il pourra vous la commander.

Procédure d'installation

Ouvrez le châssis de votre ordinateur. Cette phase est plus ou moins longue en fonction du modèle que vous possédez. Dans tous les cas, prenez votre temps et surtout repérez avec exactitude dans quel ordre les pièces ont été démontées. Par la suite, il sera plus aisé de les réassembler. Vous pouvez marquer les pièces au fur et à mesure avec du ruban adhésif.

Le processeur 68000 est facilement repérable. C'est le plus grand de la carte principale et il est signalé par une étiquette "68000" ou quelque chose d'approchant. Il va falloir l'oter.

Avant de passer à l'action, il est nécessaire de faire une mise au point importante. La plupart des processeurs 68000 sont purement et simplement introduits dans un support prévu à cet effet. Toutefois, on peut encore trouver des cartes fond de panier sur lequelles les 68000 ont été soudés. Si c'est votre cas, il y a deux façons d'agir : soit vous désoudez vous mêmes le processeur de la carte de circuit imprimé, soit vous demandez à une personne expérimentée de le faire à votre place. Attention, si vous n'avez aucune expérience en la matière, les résultats peuvent être catastrophiques. La meilleure solution est d'utiliser un bac à souder (qui réchauffe toutes les broches simultanément) de façon à ce que le processeur puisse être retiré d'un seul bloc. Pour faciliter le remplacement du processeur, utiliser le même type de support.

Supposons, pour l'instant, que vous êtes en présence d'un processeur introduit dans un support. Pour arriver à retirer le processeur, un ou deux outils sont nécessaires : des pinces spécialement conçues pour extraire le circuit imprimé d'un seul mouvement (toutes les broches en même temps), ou un tournevis. Les pinces coûtent cher; achetez les uniquement si vous désirez conserver le processeur pour un usage futur (ou si vous avez peur de l'endommager).

L'emploi d'un tournevis plat est un peu risqué, mais le résultat obtenu est le même. Insérez la lame du tournevis entre le support et l'extrémité du processeur. Faites pivoter la lame d'environ 10 pour détacher le processeur de son support. Répétez le même opération sur le côté opposé. Continuez à déplacer la lame d'une extrémité à l'autre forçant ainsi le processeur à se soulever petit à petit. Avant d'oter complètement le processeur, repérez la direction de l'encoche. Ceci est important. Enlevez ensuite le processeur à la main. Cette méthode s'applique à la quasi totalité des processeurs et tout particulièrement à ceux qui sont munis d'un nombre important de broches.

Encore un avertissement avant de poursuivre. Les composants électroniques sont extrêmement délicats. Une variation de voltage, si minime soit-elle, provoquée, par exemple, par de l'électricité statique, peut brûler un processeur (et le rendre inutilisable). C'est pourquoi, vous devez vous relier à la terre avant de manipuler le chassîs ou les processeurs.

Une fois le processeur extrait, il faut le remplacer par le nouveau. Otez le 68010 de son emballage et introduisez le dans le support qui contenait auparavant le 68000. Exercez une pression douce et uniforme sur le processeur jusqu'à ce qu'il entre parfaitement en contact avec le support.

Il est temps de remonter votre ordinateur. Assurez vous que vous n'avez égaré aucune pièce (vis, rondelles, etc...)- vous devez retrouver le nombre exact de pièces que vous avez retiré.
Vérifiez que les vis sont bien serrées. Bravo ! Vous venez de remplacer le processeur principal. Un dernier petit test de mise sous tension. Branchez l'Amiga et contrôlez toutes les connexions électriques. Mettez le sous-tension. Tout doit être normal mis à part la rapidité de la machine

Si quelque chose ne fonctionne pas correctement, deux causes possibles :

❶ Un câble ou une broche sont probablement mal connectés. C'est la première des choses à vérifier.

❷ Vos vêtements chargés d'électricité statique sont entrès en contact ave le processeur. Nous l'avons déjà mentionné, les processeurs sont très sensibles à l'électricité statique. Un simple contact de votre corps avec le processeur peut le détruire. Si aucun remède n'est efficace, vous devrez vous procurer un autre 68010. En attendant, essayez de remettre l'ancien processeur en place pour vérifier si l'ordinateur est entièrement hors d'état.

Le 68010 possède des instructions de "débogage" supplémentaires qui doivent être activées par voie logicielle. Les programmes qui débutent par l'exception 4 se planteront sur un 68010 à moins d'effectuer certaines modifications. Le disque nº 18 de Fred Fish contient le programme DeciGel qui réalise l'installation à votre place. La commande SetAlert remplit les mêmes fonctions à partir de la séquence de démarrage (vous trouverez SetAlert sur la nouvelle disquette Workbench 1.3). Profimat enfin comprend un programme équivalent, accompagné du code source.

4.4. Le ronflement du ventilateur

Vous avez un Amiga 2000 ? Au début, nous pensions que le bruit du ventilateur constituait une preuve de qualité. Au bout d'un moment, ce bruit devient franchement agaçant.

Nous vous suggérons deux solutions pour y remédier. La première nous a été soufflée par un réparateur de TV qui nous a conseillé de diminuer la puissance électrique du ventilateur. Cette proposition ne nous ayant pas vraiment séduit, nous avons opté pour une solution plus élégante.

Le ventilateur intégré à l'Amiga est un Papst Multi-Fan 8312M. Le M de la fin indique son niveau sonore. Après une petite recherche dans le commerce, nous avons déniché un modèle similaire qui présente l'avantage de faire deux fois moins de bruit. Ce modèle est le Papst Multi-Fan 8312L.

Le plus difficile dans l'installation de ce modèle, c'est de mettre la main dessus. Pour ce qui est des vis et des connexions, elles sont identiques à l'équipement d'origine. Il est vrai que le ventilateur est un peu cher : 350F environ, mais il est solide et silencieux.

A ceux qui n'ont pas les moyens de s'offrir un nouveau ventilateur, il reste la solution du réparateur de TV. Sa suggestion consiste à couper le branchement électrique positif du ventilateur et à insérer un potentiomètre 50, 5W (la gamme de régulation de tension doit être comprise entre 0 et 100). Lorsque l'on baisse le potentiomètre, le bruit s'atténue. C'est le degré d'accumulation de chaleur de votre ordinateur et non pas la vitesse du ventilateur qui doit motiver votre décision de réduire sa puissance électrique (rappelez-vous que l'accumulation de chaleur augmente avec le nombre de cartes d'extension).

Notre conclusion se présente sous la forme d'un avertissement. Le ventilateur d'Amiga est bruyant car le nombre des composants matériels à ventiler est important. Les concepteurs d'Amiga partent du principe qu'une carte est insérée dans chacun des logements disponibles, ce qui implique que le ventilateur doit tourner à sa vitesse maximale. Si ce n'est pas le cas, vous pouvez réduire sa vitesse.

4.5. Les nouveaux processeurs

La moitié des informations relatives aux nouveaux processeurs figurent dans les journaux techniques. En quoi ces processeurs peuvent-ils nous être utiles ? Avant de pouvoir répondre à cette question, passons-les en revue.

4.5.1. Le 68010 : la puissance à bas prix

Le 68010 est totalement compatible avec le jeu de commandes du 68000 qu'il reprend intégralement, mais il est plus rapide et il est doté de quatre nouvelles commandes. Ces commandes consistent en un mode boucle et 3 registres supplémentaires. Ce qui est le plus surprenant, c'est la facilité avec laquelle on peut le substituer au 68000. Pour résumer ses caractéristiques : il est bon marché, et ses dimensions et son brochage sont identiques à ceux du 68000 (cf. paragraphe 4.3).

Comment se présente l'architecture du 68010 ? Son code est exploitable par tous les assembleurs. Les trois nouveaux registres, évoqués plus haut sont les suivants :

- SourceFunctionCodeRegister (SFC)
- DestinationFunctionCodeRegister (DFC)
- VectorBaseRegister(VBR).

Ce dernier registre permet d'examiner le début de la table des vecteurs du système (comprise entre $0 et $3FF sur le 68000). Cette valeur revient à 0 après toute réinitialisation.En outre, il peut s'avérer très utile de changer tous les vecteurs par simple commutation.

Le registre Code est formé de trois bits uniquement et il permet l'accès à Read(SFC) et Write(DFC) de la même façon que les modes Utilisateur et Superviseur. Quelques nouvelles pour les fans du hardware : en connectant

les broches FC0-FC2 au bus d'adresses, vous obtenez quatre blocs de mémoire pouvant loger 16 mégaoctets. La réécriture du système d'exploitation force la séparation entre les environnements données utilisateur/ programme utilisateur et données superviseur/superviseur.

L'autre différence entre le 68000 et le 68010 réside dans le mode boucle. La technique de prélecture permet de lire une commande tandis que le processeur retient la commande précédente. Le 68000 lit la boucle suivante à partir du bus d'adresses 75000 fois et accède au bus 75000 fois. Le 68010 n'effectue cette boucle que trois fois :

```
MOVE.W #2499,DO¶
Loop: MOVE.L (AO)+, (A1)+¶
DBRA DO,Loop¶
```

Le gain de vitesse est évident. Malheureusement, seules les instructions du langage machine suivantes fonctionnent en mode boucle:

```
ABCD, ADD, ADDA, ADDX, AND, ASL, ASR, CLR, CMP,
CMPA,¶
EOR, LSL, LSR, MOVE, NBCD, NEG, NEGX, NOT, OR, ROL,
ROR,¶
ROXL, ROXR, SBCD, SUB, SUBA, SUBX, TST¶
```

Les exceptions sont un peu différentes sur le 68010 car la pile Superviseur exige un nombre supérieur de données. La zone données de la pile Superviseur étant identique sur les deux processeurs, c'est surtout la taille de la pile Supervisor du 68010 qui fait la différence. Le bit 15 du mot d'état est intéressant dans ce contexte. Il indique si le processeur exécute l'exception (0) ou s'il l'ignore et exécute la commande suivante (1). Cela signifie que les erreurs de bus et d'adresses peuvent être interceptées par voie logicielle, résolvant par la même occasion les problèmes de Gourou.

Les nouvelles commandes sont MOVEC, MOVE CCR, MOVES et RTD. La commande MOVE SR, Destination ne fonctionne qu'en mode Superviseur provoquant une méditation du Gourou. Vous pouvez programmer une routine d'exception équivalente qui permet de contourner ce problème. Peu de programmeurs le savent. Voici la syntaxe des quatre instructions (et les syntaxes de substitution) :

```
MOVEC Register, Destination¶
¶
MOVEC Source, Register¶
```

L'un des trois nouveaux registres ou l'USP peuvent être remplacés par les arguments Source et Destination. La taille des données est le mot.

```
MOVES Register, Destination¶
¶
MOVES Source, Register¶
```

MOVES réalise le transfert des données entre 4 banques de données selon les méthodes décrites ci-dessus. Il est inutile dans le cadre de manipulations matérielles majeures sur l'Amiga.

```
MOVE CCR, Destination¶
¶
MOVE CCR lit le registre d'état¶
¶
RTD Value¶
```

RTD est équivalent à RTS. Cette instruction ajoute la valeur (16 Bits) au pointeur de pile. Ceci est pratique lorsque l'on utilise la pile en tant que paramètre.

4.5.2. Le 68012 : grande capacité, bas coût.

C'est la taille du 68012 que l'on remarque en premier. Il est de forme carrée et non pas rectangulaire et ne peut par conséquent être introduit dans le support du 68000. Deux solutions s'offre à l'utilisateur :

- ➢ Installer un second support
- ➢ Acheter une carte d'adaptation

Ce circuit imprimé est compatible à 100% avec le 68010 et offre 2 gigaoctets de RAM adressable. Le premier gigaoctet réside aux adresses $0 à $3FFFFFFF et le second aux adresses $80000000- $BFFFFFFF. Le schéma suivant montre la disposition des broches:

D12	D10	D8	D7	D5	D4	D2	D1	As	|A1
D15	D14	D11	D9	D6	D3	D0	UDS	LDS	DTACK
A22	A23	D13	--	--	--	--	R/W	BG	BGACK
A21	GND	GND					--	VCC	BR
A20	VCC	A2D		MOTOTOLA			GND	GND	CLK
A19	A18	A25		68012			--	RST	HALT
A17	A15	--		(BOTTOM)			A27	VPA	VMA
A16	A12	A13	--	A28	A29	--	IPL1	IPL2	E
A14	A11	A10	A8	A5	A2	A31	FC1	IPL0	BERR
-	A9	A7	A6	A4	A3	A1	FC0	FC2	A26

Les broches représentées par des tirets ne sont pas utilisées; la broche A1 indique la partie supérieure droite du circuit imprimé.

4.5.3. Des processeurs monstres : 68020, 68030 et 6888x

68020

Les informations relatives au 68020 peuvent remplir à elles seules plusieurs volumes. Nous ne soulignerons ici que quelques points importants :

- Un bus d'adresses de 32 bits. Ce bus permet un adressage direct de 4.294.697.296 octets (environ 4 gigaoctets). La broche A0 donne accès aux adresses impaires (sur le 68000, cette même opération demande des calculs sophistiqués).

- Structure de bus dynamique : permet la permutation des bus de données en 8, 16, 24 et 32 bits.

- Arithmétique en 64 bits vraie

- 62 types d'adressage (50 d'entres eux différents)

- Accès à des bits individuels ou à des champs de bits

- 28 instructions supplémentaires

- Types de données : bits, octets, mots, double-mots, nombres décimaux codés binaire condensés, nombres décimaux codés binaire non condensés et champs de bits.

- Acceptation des instructions internes: prélecture de trois mots.

- Mémoire d'instruction interne au processeur = mémoire -cache de 256 octets

- Interface et instructions coprocesseur

- Fréquence : standard = 16MHz, autre = 24 MHz

- Trois pointeurs de pile : MasterSP, InterruptSP et UserSP

- Registre à deux mémoires-cache et SR étendu

68030/68851

Les spécifications du 68030 sont trois à quatre fois supérieures à celles qui viennent d'être décrites. Il n'en demeure pas moins compatible et dispose de 31 registres accessibles en lecture et en écriture. Le coprocesseur 68851 est fortement intégré au 68030 et fonctionne en mode multitâche au niveau matériel.

68881

Parler de puissance n'est pas un vain mot lorsqu'un coprocesseur arithmétique en virgule flottante fournit directement des calculs mathématiques à des processeurs 32/64bits. Le processeur 68881 fonctionne sur huit registres en virgule flottante et peut traiter les tailles d'opérandes suivantes :

```
Octet (8 bits)
Mot (16 bits)
Double-mot (32 bits)
Virgule flottante (32 bits)
Virgule flottante double précision (64 bits)
Virgule flottante étendue (96 bits)
Décimal codé binaire (96 bits)
```

Les fonctions mathématiques réalisent tous les calculs possibles et imaginables dont trois logarithmes différents et toutes sortes de calculs que toutes les bibliothèques mathématiques d'Amiga mises bout à bout ne suffiraient pas à effectuer. Il est pratique d'utiliser le format IEEE en virgule flottante général de façon à pouvoir réaliser des conversions.

68882

Le 68882 est une version améliorée du processeur 68881.

Chapitre 5

Workbench 1.3

Au moment où le précédent ouvrage (Trucs et Astuces) arrivait sur le marché, il n'existait que deux versions de Workbench : les versions 1.1 et 1.2. Entretemps, les équipes chargées de la conception de Commodore-Amiga ont considérablement amélioré l'interface utilisateur et AmigaDOS. Ce chapitre présente quelques unes des caractéristiques particulières qu'offre le Workbench 1.3 comme ,par exemple, l'ajout de nouveaux périphériques et de FFS (FastFileSystem). Vous y trouverez aussi de nouvelles commandes AmigaDOS et ce en quoi elles peuvent vous aider à utiliser votre ordinateur au meilleur de ses capacités.

5.1. Utilisation de Mount

Dans les précédentes versions de Workbench, les utilisateurs employaient rarement la commande Mount. Pour en savoir plus sur la commande elle-même, nous devons bien comprendre quelle est sa fonction première. La commande Mount installe un nouveau périphérique sur le système d'exploitation de l'Amiga. Examinons tout d'abord les périphériques existants. La commande Assign, non suivie de ses arguments, permet de visualiser les informations suivantes :

```
Volumes¶
¶
Tips and Tricks [Mounted]¶
RAM DISK [Mounted]¶
GFABasic¶
Workbench 1.3 F [Mounted]¶
```

```
¶
Directories¶
¶
FONTS Workbench 1.3 F:fonts¶
ENV   RAM:Env¶
T     RAM:T¶
S     Workbench 1.3 F:S¶
L     Workbench 1.3 F:L¶
C     Workbench 1.3 F:C¶
DEVS Workbench 1.3 F:devs¶
LIBS Workbench 1.3 F:libs¶
SYS   WORKBENCH 1.3 F:¶
¶
Devices:¶
¶
NEWCON DF1 DF0 PRT¶
PAR SER RAW CON RAM¶
```

Les informations classées sous Devices: indiquent les périphériques disponibles sur Workbench.

DF0: et DF1: vous sont désormais familiers. PRT: représente l'interface d'impression directe tandis que PAR: ou SER: représentent respectivement les interfaces parallèles et séries. Les sorties peuvent être dirigées sur RAW: et CON: sans pour cela accéder à Intuition. Quant à RAM:, il vous est bien connu puisqu'il représente le RAM-Disk.

Les périphériques que nous venons d'énumérer sont intégrés au système d'exploitation et accessibles à tout moment. Chaque fois que vous voulez accéder à un nouveau périphérique, vous devez signaler son existence au système par l'intermédiaire de la commande Mount. Cette méthode tient compte des améliorations futures qui seront apportées au CLI, au Shell et à l'éditeur. Pour en savoir plus sur le nouveau système d'exploitation, reportez-vous au paragraphe 5.2. (Utilisation de Workbench 1.3).

Prenons une entrée dans la liste Mount ,qui se trouve généralement dans le répertoire DEVS:. L'exemple suivant ouvre l'accès à un lecteur externe qui porte la désignation DF1: (recherchez cet exemple ou un exemple analogue dans la liste Mount).

```
DF1: Device = trackdisk.device¶
     Unit = 1¶
     Flags = 1¶
     Surfaces  = 2¶
     BlocksPerTrack = 11¶
     Reserved = 2¶
```

```
    PreAlloc = 11¶
    Interleave = 0¶
    LowCyl = 0; HighCyl = 79¶
    Buffers = 20¶
    BufMemType = 3¶
#¶
```

Les définitions débutent toujours par le nouveau nom du périphérique (DF1:) et se terminent par le signe #. Tous les éléments qui composent la définition dépendent des périphériques respectifs. Certains arguments sont fréquemment utilisés:

Device : Cet argument indique à la commande Mount le nom du périphérique. Le répertoire DEVS: est le répertoire par défaut des périphériques.

Unit : Cet argument attribue un numéro au périphérique. Le premier lecteur porte le numéro 0, le second (qu'il soit interne ou externe) porte le numéro 1 et ainsi de suite. Tous les lecteurs 5-1/4" portent le numéro 2. Cet argument est employé uniquement pour les lecteurs de disquettes.

Flags : Cet argument contient une valeur qui varie en fonction du périphérique.

Surfaces : Cet argument spécifie le nombre de faces utilisables. Sur un disque souple ces faces sont au nombre de deux. Les disques durs en comportent 4.

Reserved : Cet argument indique le nombre de blocs contenus dans le bloc d'initialisation.

BufMemType : Cet argument indique le type de mémoire utilisée par la mémoire-tampon.

```
0,1 = equal; 2,3 = CHIP-RAM; 4,5 = FAST-RAM¶
```

BootPri : Définit la priorité d'initialisation du lecteur. Les périphériques sont initialisés dans l'ordre décroissant du numéro qui leur a été affecté. Cette option est particulièrement importante dans la version 1.3 du

système d'exploitation car elle permet à l'utilisateur d'initialiser le système à partir de la diquette RAM, du disque dur ou de tout autre périphérique pouvant réaliser une opération d'initialisation. Nous n'entrerons pas dans les détails car les données dépendent fortement de chaque périphérique connecté.

5.1.1. Renommer les commandes

Si vous possédez un PC ou un compatible PC, vous aurez peut-être quelques difficultés à vous habituer aux commandes DOS de l'Amiga. Par exemple, au lieu d'entrer A: pour accéder au premier lecteur interne, il vous faut entrer cd DF0: (AmigaDOS). Ceci devient particulièrment pesant si vous passez fréquemment d'un système à un autre ou, si vous faites parti des quelques privilégiés qui ont une carte PC. Dans ce cas, nous vous conseillons de rebaptisez vos lecteurs DF0:, DF1:, etc... avec des noms compatibles IBM.

La commande Assign d'AmigaDOS permet de désigner un disque par un nouveau nom. Voici un exemple :

```
Assign B: DF1:¶
```

A présent, vous pouvez taper B: à la place de DF1:. Maintenant retirez la disquette du lecteur externe et insérez une nouvelle disquette. L'Amiga vous enjoindra d'introduire l'autre disquette. Pourquoi? Assign s'applique uniquement au répertoire et non au lecteur.

Pour résoudre cela, la solution est de copier la définition de DF1 dans la liste Mount qui contient toutes les données nécessaires à un lecteur puis de changer le nom DF1: en B:. Aprés avoir sauvegardé, il faut entrer:

```
Mount B:¶
```

Vous pouvez maintenant accéder à DF1: par son nouveau nom, B:.
Vous pouvez modifier le lecteur DF0:de façon similaire. Pour ce faire, vous devez créer une copie de l'ancienne entrée dans la liste Mount. Changez le numéro du périphérique (Unit) de 1 en 0 de façon à ce que l'accès au lecteur soit possible. Modifiez ensuite le nom en A:. Puis, entrez ce qui suit:

```
Mount A:¶
```

Les deux nouveaux périphériques sont ajoutés à la liste générale que l'on peut visualiser par la commande Assign:

```
Devices:¶
¶
A B NEWCON DF1 DF0PRT PAR SER RAW CON¶
RAM¶
```

5.1.2. Rien ne se perd, tout se transforme

Le champ d'application de la commande Mount est bien plus étendu que la simple réaffectation de périphériques. Les diverses utilisations de cette commande vous permettront d'économiser de l'argent et d'étonner vos amis. Regardons de plus prés les arguments qui accompagnent la commande Mount.

Supposons que vous ayiez acheté une boîte de 10 disquettes, sans marque, en solde pour 60,00Frs. Malheureusement, elles sont de mauvaise qualité. En formatant ces disquettes, vous vous apercevez que la première face de la plupart d'entre elles est défectueuse et que, de ce fait, la procédure de formatage s'interrompt.

Nous vous faisons part d'un petit truc: Entrez la liste Mount et copiez la définition de DF1:. Attribuez à cette copie le nom WGB:. Allez dans la liste WGB et modifiez l'argument Surfaces de 2 en 1. WGB: formatera la disquette sur une face au lieu de deux.

Entrez ce qui suit dans le CLI :

```
Format Drive WGB: Name "Surface Test"¶
```

Le formatage semble plus rapide car il ne s'applique qu'à la moitié de la disquette. Nous vous recommandons fortement de ne lire et de n'écrire sur cette disquette qu'à partir de ce même périphérique; vous aurez des difficultés à lire des disquettes monoface à partir des périphériques standard (DF0:, etc.).De même, vous ne pourrez accéder aux données que par des applications que vous aurez vous-mêmes conçues pour lire à partir WGB: (des applications normales ne vous permettront pas d'accéder aux données qui figurent sur ces disquettes). Leur grand avantage est

d'interdire toute possibilité de copie par des moyens habituels. En résumé, il est plus sage d'acheter des disquettes de la meilleure qualité possible afin d'éviter tous les problèmes liés à un formatage monoface.

La commande Mount possède une autre qualité qui sort un peu de l'ordinaire. Au cours d'une procédure de formatage, Mount intervient sur les pistes de début et de fin si celles-ci s'avèrent être endommagées. Par exemple, si vous découvrez des erreurs en lecture sur les pistes 0 à 4, entrez la liste Mount et modifiez l'argument LowCyl en 5:. Le formatage ne débutera qu'à la piste 5. Les pistes 0 à 4 ne sont pas formatées mais la procédure se poursuit normalement sur le reste de la disquette. Vous pouvez appliquer la même méthode à la fin de la disquette. Si les pistes 71 à 79 sont illisibles, modifiez simplement l'argument HighCyl en 70 et formatez comme décrit précédemment.

Un formatage expérimental peut entrainer des incompatibilités entre le Workbench et le lecteur. Le premier problème provient du Workbench. Lorsque vous connectez un nouveau lecteur externe et que vous y insérez un disque pour le formater, vous obtenez l'icône DF1:NODOS. Jusque-là pas de problème, mais un second problème demeure. Le lecteur DF1: n'est plus accessible que la disquette ait été formatée selon le procédé normal ou non. Le message "No disk in Unit 1" apparaît ce qui signifie que seul le nouveau format est autorisé. Le nouveau format est accessible aussi à partir du Workbench.

Voici quelques unes des applications intéressante de cette commande. Lorsque vous utilisez un disque de données ayant ce type de format, vous pouvez le lire sans difficultés avec vos propres programmes. Quant aux autres programmes, ils ne n'y auront pas accès sans la liste Mount appropriée.

5.2. Améliorations apportées à DOS 1.3

Les concepteurs de Commodore-Amiga ont ajouté de nombreuses améliorations et extensions à Workbench 1.3. A quelques exceptions prés, la version 1.3 est entièrement compatible avec la version 1.2. La plupart des différences résident dans les paramètres par défaut. Dans Workbench 1.2, la police de caractères par défaut était Topaz ; ce n'est pas le cas dans la version 1.3. Par conséquent, au lieu de définir l'adresse 0 pour obtenir la police Topaz, le programmeur doit définir les valeurs appropriées pour

l'ensemble des structures de données. Sur Amiga, la programmation obéit à des règles et chaque erreur commise produit un effet peu souhaitable, qu'il s'agisse de la version 1.3 ou de la version 1.2.

Les cas d'incompatibilité sont rares et ,s'ils se produisent, de simples modifications dans les paramètres permettent de les résoudre. Les nouveaux périphériques, bibliothèques et gestionnaire présentent un interêt indiscutable pour tout le monde. Les perfectionnements et les nouvelles possibilités offertes par la version 1.3 fascineront tous les utilisateurs d'Amiga qui n'auront plus besoin de s'équiper d'un Cray pour réaliser des calculs importants ou faire tourner des applications complexes. Examinons, dans un premier temps le répertoire DEVS: pour voir les améliorations qu'il nous propose.

5.2.1. Le périphérique PIPE

Le périphérique PIPE est un nouvel élément du répertoire DEVS:. Pour montrer ce dont ce gestionnaire est capable, commençons par le comparer avec un périphérique qui vous est familier : clipboard. Pour transférer des données d'une tâche à une autre à partir du CLI ou du Shell, vous utilisez clipboard qui permet leur stockage temporaire. Une fois le clipboard placé dans la liste Mount par la commande du même nom, vous pouvez introduire des données dans ce périphérique puis les en extraire.

Malheureusement, le Workbench donne lieu à de fréquente erreurs en lecture/écriture et à des retards qui se répercutent sur ce périphérique et ses données. Regardez à nouveau dans le répertoire DEVS:; vous y trouverez un répertoire presque vide . Le périphérique clipboard écrit les données dans ce répertoire et les retire si nécessaire.

L'accès au disque demande un certain temps. En ajoutant une extension à la mémoire (Le processeur 68020 dote l'Amiga d'une mémoire centrale d'une capacité de 4 294 967 296 octets -plus de 4 gigaoctets-), la plupart des données échangées sur le disque par l'intermédiaire du clipboard peuvent aussi être gérées par la RAM.

C'est à ce moment là que le périphérique PIPE entre en scène. L'opération peut se résumer à un pipeline placé dans la RAM à l'extrémité duquel des données affluent pour être ensuite déversées sur d'autres applications au moment opportun. Avant de pouvoir utiliser PIPE, il faut informer le système de son existence. Entrez

```
MOUNT PIPE:¶
```

Vous pouvez entrer la taille du pipeline dans le fichier DEVS/Mountlist au moyen de l'éditeur ED. Pour remplir le pipeline, il faut vous assurer que le périphérique NewCon reçoit les données à la place de Console. Pour ce faire, utilisez la commande (>),qui redirige la sortie du CLI:

```
DIR >PIPE:a SYS:¶
```

Le répertoire principal du disque d'initialisation est dirigé vers le pipeline. La commande ne produit aucun autre effet, si ce n'est un bref accès au disque. Pour vider PIPE et afficher son contenu à l'écran, entrez la commande suivante :

```
TYPE PIPE:a¶
```

Cette commande ne fonctionne pas si le pipeline est trop plein. Dans ce cas, la génération d'un message d'erreur rend le pipeline inutilisable.

5.2.2. Le périphérique Speak

La commande Say existait déjà dans la version 1.2 . La commande Speak de la version 1.3 a une portée entièrement différente.

Ce périphérique présente certaines similarités avec PIPE. Il se connecte sur le système et redirige les données. En revanche, Speak ne permet pas le stockage temporaire des données. Tout ce qui rentre, ressort sous forme vocale par les haut-parleurs du moniteur. Autant faire un petit essai. Entrez la commande suivante:

```
MOUNT SPEAK:¶
DIR > SPEAK: SYS:¶
```

Vous êtes fatigué de raconter des histoires à vos enfants tous les soirs, vous aimeriez bien faire une petite pause. Parfait! Amiga s'en charge. La commande suivante lance une tâche et débite le contenu du fichier MarchandDeSable:

```
RUN TYPE >SPEAK: "MarchandDeSable"¶
```

Les fanatiques de la synthèse de la parole sont invités à essayer la commande suivante après avoir introduit la disquette Extras dans df1:

```
RUN TYPE >SPEAK: df1:AmigaBASIC OPT H¶
```

Il est plus logique de rendre une invite système audible avec Ask:

```
ASK >SPEAK: "Aimez-vous Amiga?"¶
```

5.2.3. Le périphérique NewCon

Le périphérique NewCon est probablement la meilleure commande accessible à partir du Shell. Il se trouve dans la séquence de démarrage (Mount NewCon:). NewCon est analogue au périphérique Console qui ouvre les fenêtres CLI. Entrez une commande et appuyez sur <Return>. Enfoncez ensuite la touche fléchée à gauche et observez ce qui se produit sur le moniteur.

Vous pouvez à présent modifier la ligne de commande exactement de la même façon que dans la fenêtre List de AmigaBASIC. Lorsque vous avez fait une erreur de frappe dans une longue ligne de commande, vous pouvez la corriger sans avoir à la resaisir entièrement. Maintenant, entrez la commande suivante:

```
DIR DF0: DIRS¶
LIST RAM: OPT A¶
TYPE s/startup-sequence¶
```

Supposons que vous vouliez visualiser le répertoire du lecteur DF0:. A partir du CLI, il vous faudrait entrer à nouveau la commande Dir. Appuyez sur la touche fléchée vers le haut et remontez ainsi jusqu'à la commande entrée précédemment. La mémoire requise pour cette opération est limitée puisque seules les commandes sont sauvegardées. Les touches fléchées peuvent être utilisées pour rectifier les commandes.

5.2.4. Le FastFileSystem

AmigaDOS utilise beaucoup d'espace disque. Dans la version 1.3, les disques externes sont utilisés de façon intensive. Un nouveau format de disque tire parti de ces perfectionnements.

Pour vous familiariser avec la relation fichier/système, nous allons vous montrer en quoi les deux versions d'AmigaDOS diffèrent considérablement. Un disque AmigaDOS contenant des programmes est formé de blocs de 512 octets. 488 de ces 512 octets sont affectés aux données, le reste étant employé à leur gestion. Lorsque vous chargez un programme, les données affectées à la gestion exécutent un transfert de mémoire sophistiqué. C'est là que le FastFileSystem entre en action. Il élimine la nécessité des données de gestion et assure ainsi la pleine utilisation des 512 octets par les données du programme.

Lorsque plusieurs blocs d'un programme sont lus de façon séquentielle, un simple accès au disque peut réaliser cette opération. La structure de gestion de données ayant été supprimée, ce bloc peut être lu directement à partir de l'adresse voulue. La vitesse de traitement est multipliée par cinq, ce qui est considérable. L'espace disque récupéré de la sorte est aussi assez important. Nous avons calculé qu'un FastFileSystem installé sur un disque dur de 20 mégaoctets permet d'économiser 1,5 mégaoctets de mémoire.

En bref, disons que l'utilisation de FFS accroît la capacité de votre disque sans que cela vous coûte un centime.
Dans tout les cas, Workbench 1.3 offre un meilleur accès au disque.

Par défaut, le FastFileSystem est inactif. Vous devez d'abord informer le système que vous désirez utiliser le gestionnaire du même nom et à partir de quel support. Nous vous recommendons de spécifier le périphérique désiré dans la liste Mount à l'aide de ED DEV/Mountlist. Il pourra par la suite être ajouté à la séquence de démarrage avec la commande Mount. Nous avons préparé une version des listes Mount modifiée pour vous permettre de vous adapter rapidement à FFS.

5.2.5. FFS et les disques durs

Les supports de stockage les plus rentables sont les disques durs car leur capacité d'enregistrement est énorme. Nous vous donnons un exemple de liste Mount que vous pouvez intégrer dans le FastFileSystem de votre disque dur. Procédez de la façon suivante:

Copiez tous les fichiers de votre disque dur sur des disquettes. Entrez la commande suivante:

```
Mount FHD:¶
```

Formatez votre FastHardDisk à l'aide de la commande suivante :

```
FORMAT DRIVE FHD: NAME "FastHardDisk"¶
```

Copiez vos fichiers sur le périphérique FHD:. Vous serez étonné de voir le nombre de fichiers que vous pouvez placer sur le disque dur.

```
FHD:       Device = hddisk.device    /* accès au disque dur  */¶
      Filesystem = L:FastFileSystem/* mise à zéro          */¶
            Unit = 1                /* Périph. 0 en attente */¶
                                       sur AmigaDOS         */¶
           Flags = 0                /*pour Opendevice       */¶
        Surfaces = 4                /*Nb de faces du disque */¶
 BlocksperTracks = 15               /*Nb de blocs par piste */¶
        Reserved = 2                /*Blocs d'initialisation*/¶
      Interleave = 0                /*Config. du bloc       */¶
          LowCyl = 10               /*Du cylindre 10        */¶
         HyghCyl = 800              /*Au cylindre 800       */¶
         Buffers = 11               /*Buffers de lecture    */¶
      BufMemType = 1                /*Idem                  */¶
                                      (5=FastRAM)           */¶
         GlobVec = -1               /*Pas de GlobVec        */¶
           Mount = 1                /*Charge gestionnaire¶
                                      immédiatement après¶
                                      Mount                 */¶
         DosType = 0x444F5301       /*Code d'identification¶
                                      de FFS                */¶
#                                   /*Fin des entrées       */¶
```

Pour mettre en oeuvre FastHardDisk, vous devez ajouter la ligne de commande suivante à la séquence de démarrage.

```
MOUNT FHD:¶
```

5.2.6. Les nouvelles bibliothèques de fonctions mathématiques

Qui n'a jamais rêvé de posséder un processeur 68030 ou un coprocesseur "mathématique" en virgule flottante 68882 ? Le prix de ces composants n'est pas à la portée de toutes les bourses.

Les bibliothèques de fonctions mathématiques incorporées à la version 1.3 permettent de réaliser des calculs très rapidement. La vitesse de traitement des IEEE en virgule flottante, comme ceux utilisés avec les variables x#- sous BASIC, est nettement accélérée. Une analyse précise nous a permis de

déterminer que nous sommes les auteurs de la routine en virgule flottante la plus rapide jamais mise au point. Pour ne pas vous laisser en marge, voici un exemple de la façon dont cette routine peut être utilisée en BASIC :

```
DECLARE FUNCTION IEEEDPSin# LIBRARY¶
CHDIR "T&A2:bmaps"¶
LIBRARY "mathieeedoubtrans.library" 'BASIC n'accepte¶
                                    'pas les noms¶
                                    'd'accès¶
'utiliser CHDIR [chemin d'accès aux fichiers BMAP]!¶
PI#=4*ATN(1)   'PI est calculé (cercle complet=2*PI)¶
CIRC#=2*PI#¶
FOR I%=1 TO 359                    'cercle en degrés¶
 ANGLE#=CIRC#/I%                   'angle from 2*PI¶
 HighLong&=PEEKL(VARPTR(ANGLE#))   'premier Long ¶
                                   'd'une virgule¶
                                   'flottante
double¶
                                   'précision¶
 LowLong& =PEEKL(VARPTR(ANGLE#)+4) 'second Long
d'une¶
                                   'virgule¶
                                   'flottante¶
 SINUS#=IEEEDPSin#(HighLong&,LowLong&)'Appel
fonction¶
 PSET(I%,90-INT(SINUS#*50)),1       'Dessine pixel¶
NEXT¶
FOR I%=1 TO 359                   'Demo uniquement¶
 ANGLE#=CIRC#/I%                  'Voyez comme BASIC¶
 SINUS#=SIN(ANGLE#)               'est rapide...¶
 PSET(I%,90-INT(SINUS#*50)),2     'autre couleur¶
NEXT¶
```

La première boucle FOR/NEXT est plus lente que la seconde. La fonction VARPTR doit être invoquée 720 ,PEEKL 720 fois également, l'addition de la valeur 4, 360 fois, et les routines 36O fois avec affectation de deux valeurs longues.Tout cela n'est guère rapide. L'ajout de variables et de lignes faussent la première boucle. Toutes ces limitations sont impressionnantes. Et pourtant, la première boucle est aussi rapide que la seconde.

La bibliothèque mathématique MathIEEEdoubtrans comprend toutes les fonctions mathématiques transcendantes qu'il est possible d'exécuter sur des nombres à virgule flottante en double précision. La bibliothèque MathIEEEdoubtrans, qui comprend des fonctions de calcul simples, est rapide. Les fonctions mathématiques transcendantes sont pratiques pour les programmeurs BASIC car elles permettent de calculer un arc sinus aussi rapidement qu'un sinus sans utiliser de programme de calcul. Voici une vue d'ensemble des différentes fonctions:

```
MathIeeeDoubBas-Library¶
x,y,Double                          Nb à virgule flottante¶
                                    en double précision
(BASIC: 2 longs au lieu             de x et y).¶
Long                                Entier long positif/négatif¶
Long    IEEEDPFix       (x)         Conversion entier¶
(D0)    -30     (DO/D1)             long/virgule flottante¶
                                    double précision¶
Double  IEEEDPFlt   (Long)          Conversion virgule
(D0/D1) -36                         flottante double précision/entier
                                    long¶
Long    IEEEDPCmp       (x,y)       Compare x et y¶
(DO)    -42     (D0/D1,D2/D3)       (cc mis pour bcc)¶
                                    S'applique à :¶
                                    x > y-->1¶
                                    x=y  -->0¶
                                    x<y -->-1¶
Long    IEEEDPTst       (x)         Compare x et 0¶
(DO)    -48     (DO/D1)             (cc positionné;¶
                                    résultat traité¶
                                    comme dans IEEEDPCmp¶
                                    quand y = 0)¶
Double  IEEEDPAbs       (x)         Renvoie la valeur¶
(DO/D1) -54     (DO/D1)             absolue de x¶
Double  IEEEDPNeg       (x)         Fonction:¶
(DO/D1) -60     (DO0/D1)            Double=-x¶
¶
MathIeeeDoubTrans-Library¶
Double  IEEEDPAdd       (x,y)       Fonction:¶
(DO/D1) -66     (D0/D1,D2/D3)       Double=x+Y¶
Double  IEEEDPSub       (x,y)       Fonction:¶
(D0/D1) -72     (D0/D1,D2/D3)       Double=x-y¶
Double  IEEEDPMul       (x,y)       Fonction:¶
(D0/D1) -78     (D0/D1,D2/D3)       Double=x*y¶
Double  IEEEDPDiv       (x,y)       Fonction:¶
(D0/D1) -84     (D0/D1,D2/D3)       Double=x/y¶
Double  IEEEDPFloor     (x)         Renvoie le plus¶
(DO/D1) -90     (D0/D1)             grand entier inférieur¶
                                    ou égal à x.¶
Double  IEEEDPCeil      (x)         Renvoie le plus¶
(D0/D1) -96     (D0/D1)             petit entier supérieur¶
                                    ou égal à x.¶
Double  IEEEDPAtan      (x)         Renvoie l'arc tangente¶
(D0/D1) -30     (D0/D1) de x.¶
Double  IEEEDPSin       (x)         Renvoie le sinus de x¶
(D0/D1) -36     (D0/D1)¶
Double  IEEEDPCos       (x)         Renvoie le cosinus de x¶
(D0/D1) -42     (D0/D1)¶
Double  IEEEDPTan       (x)         Renvoie la tangente de x¶
(D0/D1) -48     (D0/D1)¶
Double  IEEEDPSincos(x,VARPTR)      Double calcul:¶
```

```
(D0/D1) -54      (D0/D1,A0)        Calcule le sinus de x,¶
                                   place le cosinus de x dans VARPTR.¶
Double  IEEEDPSinh       (x)       Renvoie le sinus¶
(D0/D1) -60      (D0/D1)           hyperbolique de x¶
Double  IEEEDPCosh       (x)       Renvoie le cosinus¶
(D0/D1) -66      (D0/D1)           hyperbolique de x¶
Double  IEEEDPTanh       (x)       Renvoie la tangente¶
(D0/D1) -72      (D0/D1)           hyperbolique de x¶
Double  IEEEDPExp        (x)       Exposant de e¶
(D0/D1) -78      (D0/D1)           Fonction:¶
                                   Double=e^x¶
Double  IEEEDPLog        (x)       Renvoie le logarithme¶
(D0/D1) -84         (D0/D1)        naturel de x¶
Double  IEEEDPPow        (x,y)     Fonction:¶
(D0/D1) -90      (D0/D1,D2/D3)     Double=x^y¶
Double  IEEEDPTSqrt      (x)       Renvoie la racine¶
(D0/D1) -96      (D0/D1)           carrée de x¶
Float   IEEEDPTieee      (x)       Calc. x en IEEE à¶
(D0)    -102     (D0/D1)           virgule flottante¶
                                   en double précision¶
Double  IEEEDPFieee      (Float)   Calcule les nombre en¶
(D0/D1) -108     (D0)              virgule flottant simple¶
                                   précision en double précision¶
Double  IEEEDPAsin       (x)       Renvoie l'arcsinus de x¶
(D0/D1) -114     (D0/D1)¶
Double  IEEEDPAcos       (x)       Renvoie l'arc cosinus¶
(D0/D1) -120     (D0/D1)           de x¶
Double  IEEEDOLog10      (x)       Renvoie le logarithme en¶
(D0/D1) -126     (D0/D1)           base 10 de X¶
```

Le programmeur BASIC ne doit pas oublier que les nombres en virgule flottante double précision ne sont pas autorisés dans les routines système. Cette variable de 64 bits doit être définie sous la forme de deux variables de 32 bits qui obtiennent leur valeur par l'intermédiaire de PEEKL (VARPTR()). La démonstration sur les sinus illustre cette technique. Une autre caractéristique de BASIC est la brièveté de l'intitulé de la bibliothèque IEEE correspondant à la commande du même nom.

Des noms d'accès plus longs peuvent être attribués à la plupart des bibliothèques. Dans le cas qui nous concerne, le nom d'accès est déjà suffisamment long pour qu'il soit possible d'y ajouter encore des indications. Un nom d'accès disproportionné débouche sur l'erreur File not found. Pour accéder aux bibliothèques mathématiques, il faut placer la disquette de démarrage du système, accompagnée des fichiers BMAP correspondants, dans le répertoire courant ou dans le répertoire LIBS:, ou encore ajouter l'instruction CHDIR avant la commande Library.

Chapitre 6

Le printer.device

Pour utiliser l'imprimante connectée à votre Amiga, vous avez besoin du printer.device. Pour peu que vous possédiez le driver d'imprimante adéquat et que le printer.device soit disponible, la communication entre l'imprimante et l'ordinateur se fait parfaitement et avec un minimum de tracas.

Ce chapitre montre comment définir votre imprimante de façon à réaliser des opérations qui sortent un peu des sentiers battus. Vous y trouverez un programme permettant de contrôler votre imprimante sans passer par Preferences, ainsi qu'un programme permettant de réaliser facilement des impressions à partir d'une fenêtre.

6.1. Sélection des paramètres d'impression

Lorsque l'on feuillette un magazine d'informatique, on est litéralement assailli de publicités alléchantes vantant les mérites d'une bonne centaine de modèles d'imprimantes. La gamme de prix proposée est étendue tandis que la technique et la qualité d'impression varient d'une imprimante à l'autre.

Chaque type d'imprimante présente à la fois des avantages et des inconvénients dont nous dressons la liste ci-dessous:

➢ Les imprimantes thermiques sont bon marché et silencieuses, mais elles nécessitent l'usage d'un papier spécial et ne permettent pas toujours l'impression graphique.

➢ Les imprimantes à marguerite sont parfaites pour imprimer du courrier, des thèses, etc..., mais elles ne permettent pas l'impression de graphiques à l'exception de quelques graphiques tout à fait rudimentaires, réalisés à l'aide des caractères disponibles.

➢ Les imprimantes matricielles produisent des graphiques (même en couleur). Cependant, le mode NLQ (qualité courrier) est souvent insuffisant pour réaliser des documents de qualité professionnelle.

➢ Les imprimantes à laser allient rapidité d'impression, haute résolution et production de graphiques. Toutefois leur prix se situe hors de portée de l'utilisateur moyen.

➢ Les imprimantes à jet d'encre sont silencieuses, elles possèdent une bonne résolution mais leur qualité graphique varie considérablement

Chacun des types d'imprimantes que nous venons de décrire répond au moins à l'une de vos attentes en matière d'impression.

Une fois l'imprimante correcte sélectionnée, dans l'écran Change Printer de Preferences (sur la disquette Workbench), Amiga convertit automatiquement les commandes d'impression générales ainsi que les codes de contrôle spécifiques à l'imprimante afin de les adapter à l'imprimante que vous utilisez. Ces codes ont pour fonction de rendre vos programmes compatibles ou presque avec votre imprimante.

Généralement, Preferences gère lui même les codes de contrôle. Toutefois, vous pouvez les annuler et les remplacer à l'aide du programme ci-dessous. Ce programme vous donnera une idée de la façon dont le driver d'imprimante communique avec votre imprimante. Il montre aussi comment adapter le driver à vos propres besoins.

Les contraintes liées à la mise en page du présent ouvrage ont entrainé la coupure de certaines lignes de BASIC en deux. Les caractères (P) qui figurent dans le programme ont été ajoutés uniquement pour marquer la fin d'une ligne de programme et, à ce titre, ne doivent pas être reproduits.

```
'************************************************¶
'* Programme: Lecture des paramètres d'impression¶
'* Date: 28 Mai 88¶
'* Auteur: tob¶
'* Version: 1.3¶
'************************************************¶
CLS¶
PRINT "Recherche des fichiers .bmap!"¶
'EXEC-LIBRARY¶
DECLARE FUNCTION AllocMem& LIBRARY¶
DECLARE FUNCTION DoIO& LIBRARY¶
DECLARE FUNCTION OpenDevice% LIBRARY¶
DECLARE FUNCTION AllocSignal% LIBRARY¶
DECLARE FUNCTION FindTask& LIBRARY¶
LIBRARY "t&a2:bmaps/exec.library"¶
init:       '¶
            GetPrinterData¶
¶
            PRINT "Nom de l'imprimante       : ";
prt.name$¶
            PRINT "Type de l'imprimante      : ";
prt.typ$¶
            PRINT "Gamme de couleurs         : ";
prt.colour$¶
            PRINT "Caractères par ligne      : ";
prt.columns%¶
            PRINT "Nb de jeux de caractère   : ";
prt.charsets%¶
            PRINT "Nb de lignes horiz.       : ";
prt.rows&¶
            PRINT "Nb max. de points horiz.  : ";
prt.xdots&¶
            PRINT "Nb max de points vert.    : ";
prt.ydots&¶
            PRINT "Densité: Points/pouces h. : ";
prt.xdotspi&¶
            PRINT "Densité: Points/poouces v.: ";
prt.ydotspi&¶
¶
            END¶
¶
SUB GetPrinterData STATIC¶
      SHARED prt.DRPReq&¶
      SHARED prt.typ$, prt.colour$, prt.name$¶
      SHARED prt.columns%, prt.charsets%¶
      SHARED prt.rows&, prt.xdots&, prt.ydots&¶
      SHARED prt.xdotspi&, prt.ydotspi&¶
¶
      DIM prt.color$     (9)¶
      DIM prt.printer$   (3)¶
¶
      prt.color$   (1) = "Noir-Blanc"¶
```

```
      prt.color$   (2) = "Jaune-Magenta-Cyan"¶
      prt.color$   (3) = "Jaune-Magenta-Cyan ou Noir-
Blanc"¶
      prt.color$   (4) = "Jaune-Magenta-Cyan-Noir"¶
      prt.color$   (5) = "Bleu-vert-Rouge-Blanc"¶
      prt.color$   (6) = "Noir-Blanc Invers"¶
      prt.color$   (7) = "Bleu-Vert-Rouge"¶
      prt.color$   (8) = "Bleu-Vert-Rouge ou Noir-
Blanc"¶
      prt.color$   (9) = "Bleu-Vert-Rouge-Blanc"¶
¶
      prt.printer$(0) = "Imprimante Texte N/B"¶
      prt.printer$(1) = "Imprimante graphique N/B"¶
      prt.printer$(2) = "Imprimante Texte couleur"¶
      prt.printer$(3) = "Imprimante graphique
couleur"¶
¶
¶
      OpenPrinter¶

¶
      prt.printerdata&  = PEEKL (prt.DRPReq& + 20)¶
      prt.extendeddata& = (PEEKL (prt.printerdata& +
92) + 12)¶
      prt.name$         = ""¶
      prt.name&         = PEEKL (prt.extendeddata&)¶
      prt.printer%      = PEEK  (prt.extendeddata& +
20)¶
      prt.color%        = PEEK  (prt.extendeddata& +
21)¶
      prt.columns%      = PEEK  (prt.extendeddata& +
22)¶
      prt.charsets%     = PEEK  (prt.extendeddata& +
23)¶
      prt.rows&         = PEEKW (prt.extendeddata& +
24)¶
      prt.xdots&        = PEEKL (prt.extendeddata& +
26)¶
      prt.ydots&        = PEEKL (prt.extendeddata& +
30)¶
      prt.xdotspi&      = PEEKW (prt.extendeddata& +
34)¶
      prt.ydotspi&      = PEEKW (prt.extendeddata& +
36)¶
¶
      prt.typ$          = prt.printer$
(prt.printer%)¶
      prt.colour$       = prt.color$   (prt.color%)¶
¶
      count = NULL¶
      char  = PEEK (prt.name& + count)¶
¶
      WHILE char <> NULL¶
```

```
        prt.name$ = prt.name$ + CHR$ (char)¶
        count     = count + 1¶
        char      = PEEK (prt.name& + count)¶
      WEND¶
¶
      ClosePrinter¶
END SUB¶
SUB OpenPrinter STATIC¶
      SHARED mem.chunk&¶
      SHARED prt.DRPReq&¶
¶
      mem.clear&  = 2^16          'Mise à zéro de la
mémoire¶
      mem.DRPReq% = 62            '62 octets pour
DRPStructure¶
      mem.port%   = 37            '37 octets pour
Port-Struct.¶
      mem.label%  = 4             '4  octets for
l'organization¶
      mem.size%   = mem.DRPReq% + mem.port% +
mem.label%¶
¶
      mem.chunk&  = AllocMem& (mem.size%,
mem.clear&)¶
      IF mem.chunk& = NULL THEN¶
        ERROR 7                   'OUT OF MEMORY
ERROR¶
      END IF¶
¶
      prt.label&  = mem.chunk&¶
      prt.DRPReq& = mem.chunk& + mem.label%¶
      prt.port&   = mem.chunk& + mem.label% +
mem.DRPReq%¶
      prt.name$   = "printer.device" + CHR$(0)¶
¶
      POKEL prt.label&, mem.size% 'allocate memory
size¶
¶
      status% = OpenDevice% (SADD(prt.name$), 0,
prt.DRPReq&, 0)¶
      IF status% <> NULL THEN¶
        PRINT "L'imprimante n'est pas disponible."¶
        CALL FreeMem (mem.chunk&, mem.size%)¶
        EXIT SUB¶
      END IF¶
END SUB¶
¶
SUB ClosePrinter STATIC¶
      SHARED mem.chunk&¶
¶
      mem.size%   = PEEKL (mem.chunk&)¶
      prt.DRPReq& = mem.chunk& + 4¶
      CALL CloseDevice (prt.DRPReq&)¶
```

```
      CALL FreeMem (mem.chunk&, mem.size%)¶
END SUB¶
```

Variables

```
prt.DRPReq&          Structure DumpRastPort d'E/S
                     (adresse de départ)
prt.typ$             Type d'imprimante
prt.colours$         Gamme de couleurs
prt.name$            Nom de l'imprimante
prt.columnss%        Nombre de caractères par ligne
prt.charsets%        Nombre de polices de caractères
                     disponibles
prt.rows&            Nombre d'aiguilles de la tête
                     d'impression
prt.xdots&           Nombre maximum de pixels
                     horizontaux
prt.ydots&           Nombre maximum de pixels verticaux
prt.xdotspi&         Densité horizontale
                     (points/pouces)
prt.ydotspi&         Densité verticale (points/pouces)

GetPrinterData():

prt.color$()              Tableau- types de couleurs
prt.printer$()            Types d'imprimantes
prt.printerdata&          Adresse de départ,
                          structure PrinterData
prt.extendedddata&        Adresse de départ,
                          structure ExtendedData
prt.name&                 Adresse de départ, chaîne
                          du nom
prt.printer%              Numéro de code du type
                          d'imprimante
prt.color                 Numéro de code du type de
                          couleur
count                     Compteur
char                      Caractère de lecture

OpenPrinter:

memchunk&            Adresse de départ, mémoire
                     réservée
memclear&            =2^16, met à 0 la mémoire
                     disponible
mem.DRPReq%          =62, réserve 62 octets pour la
                     structure
memport%             =38, réserve 38 octets pour la
                     structure
```

```
mem.label%          =4; réserve 4 octets pour
                    l'organisation
mem.size%           Besoins en mémoire en octets
prt.label&          Adresse de départ, étiquette
prt.DRPReq&         Adresse de départ, structure
                    DumpRastPort
prt.port&           Adresse de départ, structure
                    Port
prt.name$           Nom du périphérique
status%             0 = Fonctionnement correct
```

Description du programme

En y regardant de plus près, l'on s'aperçoit que le programme ci-dessus est composé de trois sous-programmes :

```
GetPrinterData¶
OpenPrinter¶
ClosePrinter¶
```

Le sous-programme GetPrinterData présente un grand intérêt pour l'utilisateur. Il appelle ,de façon interne, les deux autres sous-programmes qui vont chercher les informations qui leur sont nécessaires dans la structure PrinterExtendedData. Avant de parvenir à cette étape, il est indispensable d'ouvrir un accès à l'imprimante à travers le printer.device. C'est le sous-programme OpenPrinter qui est à l'origine de cette opération.

Ensuite, la fonction Allocmem() de Exec alloue un espace mémoire aux deux structures Port et DumpRastPort. En outre, AllocMem() réserve 4 octets destinés au stockage de la capacité de la mémoire absolue listée pour FreeMem().

Par cette méthode, la fonction OpenDevice() de Exec ouvre un accès à l'imprimante. Cet appel renvoie un compte-rendu Status au système. Tant que la valeur de Status est différente de zéro, l'accès à l'imprimante ne peut être obtenu.

Pour quelle raison cette valeur est-elle différente de zéro ? Soit une autre tâche accède au même moment à l'imprimante, soit l'accès à l'imprimante n'a pas été correctement fermée auparavant.

Lorsqu'un accès à l'imprimante est ouvert, la structure DumpRastPort contient un pointeur sur la structure PrinterData. Quand le pointeur est réinitialisé, il pointe sur la structure de données PrinterExtended où les données nécessaires sont enregistrées.

Les données sont lues et enregistrées dans les variables correspondantes. Ensuite, l'imprimante est refermée par un appel à la routine ClosePrinter. Cette étape est obligatoire. Si un accès à l'imprimante ouvert par un programme n'est pas refermé par le même programme, l'accès à l'imprimante sera interdit jusqu'à une nouvelle initialisation de l'ordinateur.

Voici un exemple du résultat du programme :

```
Nom de l'imprimante          : EpsonQ¶
Type de l'imprimante         : Graphique couleur¶
Gamme de couleurs            :
Jaune-Magenta-Cyan-Noir¶
Caractères par ligne         : 80¶
Nombre de jeux de caract.    : 10¶
Nombre de lignes horiz       : 24¶
Nombre de points horiz. max  : 720¶
Nombre de points vert. max   : 0¶
Densité Points/pouces horiz  : 90¶
Densité Points/pouces vert   : 180¶
```

6.2. Les impressions graphiques au moyen du printer.device

Le programme suivant est un exemple de programme de contrôle d'impression. Il présente les éléments essentiels permettant d'imprimer le contenu d'une fenêtre BASIC sous la forme d'une impression graphique ou d'une copie d'écran.

Ce programme prend en charge l'ensemble des indicateurs spéciaux incorporés au système d'exploitation 1.3. Ces indicateurs vous permettent de réduire la taille d'une fenêtre, de l'agrandir, de modifier sa structure, de la centrer,etc...

Les contraintes liées à la mise en page du présent ouvrage ont entrainé la coupure de certaines lignes de BASIC en deux. Les caractères (¶) qui figurent dans le programme ont été ajoutés uniquement pour marquer la fin d'une ligne de programme et à ce titre ne doivent pas être reproduits.

```
'************************************¶
'* Program: Impression graphique¶
'* Date: 28 Mai 1988¶
'* Auteur: tob¶
'* Version: 1.3¶
'************************************¶
PRINT "Recherche des fichiers .bmaps!"¶
'EXEC-LIBRARY¶
DECLARE FUNCTION AllocMem& LIBRARY¶
DECLARE FUNCTION DoIO& LIBRARY¶
DECLARE FUNCTION OpenDevice% LIBRARY¶
DECLARE FUNCTION AllocSignal% LIBRARY¶
DECLARE FUNCTION FindTask& LIBRARY¶
LIBRARY "T&A2:bmaps/exec.library"¶
init:   '¶
   CIRCLE (100,100),100¶
   PRINT STRING$ (100,"_")¶
¶
   special.nothing    = 0 'Pas d'effets spéciaux¶
   special.milcols    = 1 'Impresion horiz. par 1/100
'0 Pouces¶
   special.milrows    = 2 'Impression vert. par 1/100
'0 Pouces¶
   special.fullcols   = 4 'Largeur maximale  'g¶
   special.fullrows   = 8 'Hauteur maximale  'g¶
   special.fraccols   =   16 'fraction de la largeur
maximale¶
   special.fracrows   =   32 'idem pour la hauteur¶
   special.center     =   64 'Graphique centré sur la
page¶
   special.aspect     =  128 'correction du rapport
larg./haut.¶
   special.density1   =  256 'Position 1 (faible)¶
   special.density2   =  512 'Position 2¶
   special.density3   =  768 'Position 3¶
   special.density4   = 1024 'Position 4¶
   special.density5   = 1280 'Position 5¶
   special.density6   = 1536 'Position 6¶
   special.density7   = 1792 'Position 7 (forte)¶
   special.noformfeed= 2048 'Pas d'avance papier¶
   special.trustme    = 4096 'Pas de réinitialisation¶
   special.noprint    = 8096 'Calcul uniquement,pas
d'impression¶
¶
   Hardcopy (special.center + special.density4),
100&, 100&¶
```

```
¶
   'pour imprimante noir/blanc, écran noir et blanc¶
¶
   PALETTE 0,1,1,1¶
   PALETTE 1,0,0,0¶
¶
   Hardcopy (special.aspect + special.fullcols +
special.fullrows), 0&, 0&¶
¶
¶
   END¶
¶
SUB Hardcopy (flags, x&, y&) STATIC¶
   SHARED prt.DRPReq&¶
¶
   OpenPrinter¶
¶
   POKEL prt.DRPReq& + 52, x&¶
   POKEL prt.DRPReq& + 56, y&¶
   POKEW prt.DRPReq& + 60, flags¶
   InitDRPReq¶
¶
   PrtErr% = DoIO& (prt.DRPReq&)¶
¶
   PrtErr$  (0)    = "PAS D'ERREUR."¶
   PrtErr$  (1)    = "IMPRESSION INTERROMPUE PAR
L'UTILISATEUR."¶
   PrtErr$  (2)    = "L'IMPRIMANTE N'EST PAS
GRAPHIQUE."¶
   PrtErr$  (3)    = "./."¶
   PrtErr$  (4)    = "TAILLE D'IMPRESSION INVALIDE"¶
   PrtErr$  (5)    = "./."¶
   PrtErr$  (6)    = "PAS DE MEMOIRE POUR LES
VARIABLES INTERNES."¶
   PrtErr$  (7)    = "PAS DE MEMOIRE POUR LE BUFFER
D'IMPRESSION."¶
¶
   result$   = PrtErr$ (PrtErr%)¶
¶
   PRINT result$¶
¶
   ClosePrinter¶
END SUB¶
SUB OpenPrinter STATIC¶
   SHARED mem.chunk&¶
   SHARED prt.DRPReq&¶
   SHARED prt.port&¶
¶
   mem.clear&  = 2^16 'Mise à zéro de la mémoire¶
   mem.DRPReq% = 62   '62 octets pour Structure
DumpRastport¶
   mem.port%   = 38   '38 octets for Port-Structure¶
```

```
   mem.label%   = 4     '4  octets pour l'organisation¶
   mem.size%    = mem.DRPReq% + mem.port% +
mem.label%¶
¶
   mem.chunk&   = AllocMem& (mem.size%, mem.clear&)¶
   IF mem.chunk& = NULL THEN¶
  ERROR 7        'OUT OF MEMORY ERROR¶
   END IF¶
¶
   prt.label&   = mem.chunk&¶
   prt.DRPReq&  = mem.chunk& + mem.label%¶
   prt.port&    = mem.chunk& + mem.label% +
mem.DRPReq%¶
   prt.name$    = "printer.device" + CHR$(0)¶
¶
   POKEL prt.label&, mem.size% 'allocate  memory
size¶
¶
   status% = OpenDevice% (SADD(prt.name$), 0,
prt.DRPReq&, 0)¶
   IF status% <> NULL THEN¶
  PRINT "Printer is not free."¶
  CALL FreeMem (mem.chunk&, mem.size%)¶
  EXIT SUB¶
   END IF¶
END SUB¶
SUB InitDRPReq STATIC¶
   SHARED prt.DRPReq&¶
   SHARED prt.port&¶
   SHARED p.sigBit%¶
¶
   w.window&     = WINDOW(7)¶
   w.rastport&    = PEEKL (w.window& + 50)¶
   w.width%      = PEEKW (w.window& + 112)¶
   w.height%       = PEEKW (w.window& + 114)¶
   w.screen&      = PEEKL (w.window& + 46)¶
   w.viewport&    = w.screen& + 44¶
   w.colormap&    = PEEKL (w.viewport& + 4)¶
   w.vp.modi%     = PEEKW (w.viewport& + 32)¶
¶
   p.sigBit% = AllocSignal%(-1)¶
   IF p.sigBit% = -1 THEN¶
  PRINT "No Signalbit free!"¶
  CALL FreeMem(p.io&,100)¶
  EXIT SUB¶
   END IF¶
   p.sigTask& = FindTask&(0)¶
¶
   POKE  prt.port&+8,4¶
   POKEL prt.port&+10,prt.port&+34¶
   POKE  prt.port&+15,p.sigBit%¶
   POKEL prt.port&+16,p.sigTask&¶
   POKEL prt.port&+20,prt.port&+24¶
```

```
   POKEL prt.port&+28,prt.port&+20¶
   POKE  prt.port&+34,ASC("P")¶
   POKE  prt.port&+35,ASC("R")¶
   POKE  prt.port&+36,ASC("T")¶
¶
   CALL AddPort(prt.port&)¶
¶
   POKE  prt.DRPReq& +  8, 5¶
   POKEL prt.DRPReq& + 14, prt.port&¶
   POKEW prt.DRPReq& + 28, 11¶
   POKEL prt.DRPReq& + 32, w.rastport&¶
   POKEL prt.DRPReq& + 36, w.colormap&¶
   POKEL prt.DRPReq& + 40, w.vp.modi%¶
   POKEW prt.DRPReq& + 48, w.width%¶
   POKEW prt.DRPReq& + 50, w.height%¶
¶
   IF PEEKL (prt.DRPReq& + 52) = 0 THEN¶
  POKEL prt.DRPReq& + 52, x&¶
   END IF¶
¶
   IF PEEKL (prt.DRPReq& + 56) = 0 THEN¶
  POKEL prt.DRPReq& + 56, y&¶
   END IF¶
END SUB¶
¶
SUB ClosePrinter STATIC¶
   SHARED mem.chunk&¶
   SHARED prt.port&¶
   SHARED p.sigBit%¶
¶
   mem.size%   = PEEKL (mem.chunk&)¶
   prt.DRPReq& = mem.chunk& + 4¶
   CALL CloseDevice (prt.DRPReq&)¶
   CALL RemPort  (prt.port&)¶
   CALL FreeSignal  (p.sigBit%)¶
   CALL FreeMem  (mem.chunk&, mem.size%)¶
END SUB¶
```

Variables

PrtErr%	Nombre d'erreurs de la procédure d'E/S
PrtErr$()	Message d'erreur
result$	Message d'erreur courant

Description du programme

Comme vous l'avez probablement déjà remarqué, ce programme comprend les sous-programmes OpenPrinter et ClosePrinter décrits dans le paragraphe 6.1. Les sous-programmes HardCopy et InitDRPReq sont nouveaux. Le programme HardCopy présente un grand interêt pour les utilisateurs. Il a pour fonction de transférer la fenêtre BASIC courante vers l'imprimante sous forme graphique puis d'appeler les autres sous-programmes.

Avant d'imprimer un écran graphique, un accès à l'imprimante doit être obtenu. Le programme OpenPrinter ouvre un accès à l'imprimante exactement de la même façon que dans le programme présenté dans le paragraphe 6.1. Le programme écrit par un POKE la largeur et la hauteur de l'image à imprimer dans la structure DumpRastPort. La même chose se produit avec les bits spéciaux.

Le programme appelle ensuite InitDRPReq. Cette routine remplit le reste de la structure avec des valeurs standards, puis elle revient à la fenêtre BASIC.

Au moment voulu, la fonction DoIO& de Exec transfère la structure IORequest à l'imprimante. Si l'imprimante s'arrête, ou si la commande ne peut être exécutée pour une raison quelconque, cette fonction renvoie un code d'erreur à la variable Status%. Le programme convertit ce code d'erreur en un texte compréhensible et l'affiche à l'écran. La routine ClosePrinter ferme l'accès à l'imprimante et le programme s'achève.

6.2.1. Application de Hardcopy

La fonction HardCopy a un caractère étonnamment universel. Elle utilise toutes les fonctions offertes par le printer.device. L'appel au sous-programme ressemble à la séquence qui suit:

```
Hardcopy flags, width&, height&¶
     flags: indicateurs spéciaux¶
     height: hauteur de l'impression¶
     width: largeur de l'impression¶
```

Indicateurs :

```
special.nothing
```

L'impression ne comporte aucun effet spécial

```
special.milcols
```

La largeur d'impression se fait par 1/1000 de pouces au lieu de points (1 pouce équivaut à 2.5cm)

```
Hardcopy special.milcols, 9000, 400
```

Cet appel imprime un graphique de la taille spécifié dans les arguments. Par exemple, dans la commande ci- dessus, la largeur par défaut est de 9 pouces (22.5 cm) et la hauteur de 400 points.

```
special.milrows
```

Identique à special.milcols mais pour la hauteur

```
special.fullcols
```

L'impression est aussi large que le matériel le permet, quelle que soit la valeur donnée en argument.

```
special.fullrows
```

Identique à special.fullcols mais pour la hauteur

```
special.fraccols
```

La largeur spécifiée est interprétée à X/65535ème de la largeur maximum.

```
special.fracrows
```

Identique à special.fraccols. La largeur spécifiée est interprétée à x/65535ème de la largeur maximum.

```
special.center
```

Le programme imprime le graphique centré dans la page. L'indicateur special.center ignore tous les paramètres de dimensions spécifiés précédemment.

```
special.aspect
```

Cet indicateur conserve le rapport hauteur/largeur, quelles que soient les modifications apportées par l'utilisateur.

```
special.density1-7(V1.3)
```

Densité d'impression 1= faible (par défaut)
7= forte

```
special.noformfeed(V1.3)
```

Inactive l'avance papier. Particulièrement utile si on utilise une imprimante laser. Permet à l'utilisateur d'intégrer du texte et des graphiques.

```
special.trustme
```

Aucune réinitialisation n'est envoyée à l'imprimante

```
special.noprint(V1.3)
```

Traite toutes les descriptions et calcule toutes les dimensions sans réaliser d'impression. Cette commande permet à l'utilisateur de vérifier les paramètres avant de lancer une impression.

Chapitre 7

Les disquettes Workbench et Extras

La disquette Workbench contient toutes les données necéssaires à la réalisation d'applications d'ordre général. On y trouve les fichiers relatifs aux polices de caractères, aux drivers d'imprimante, aux commandes CLI et aux bibliothèques du système. La nouvelle disquette Workbench a été dotée de programmes utilitaires et améliorée. De même, la disquette Extras contient bien plus que de simples programmes en BASIC.

Ce chapitre présente toutes les améliorations apportées au Workbench dans le souci de faciliter et d'optimiser vos sessions sous AmigaDOS. Toutes les facettes du programme Preferences y sont explorées. Vous apprendrez comment redéfinir certains des paramètres de Preferences par programmation et vous y trouverez un tableau énumérant l'ensemble de ces paramètres commentés en langage C.

Le paragraphe 7.2 est consacré à l'étude de quelques aspects de la disquette Extras méconnus de l'utilisateur et qui l'aideront à en faire un meilleur usage.

7.1. Preferences

Parmi l'éventail de programmes fournis avec l'Amiga, Preferences est sans aucun doute le plus important. Son importance réside dans la maîtrise qu'il permet d'exercer sur l'environnement de travail. Preferences crée et gère tous les paramètres que l'utilisateur sélectionne pour mettre au point un environnement de travail adapté à ses besoins (couleurs, graphisme du

pointeur, horloge, etc... pour ne citer que quelques exemples). L'un des inconvénients de Preferences provient de la taille du programme (55K). Le temps de chargement requis par le programme est un facteur propre à dissuader de nombreux utilisateurs.

Les moyens de contourner ce problème ne manquent pas. L'une des options possibles est de copier Preferences sur la disquette RAM pour en accélérer l'accès. Les toutes dernières versions de Workbench (y compris 1.3) sont équipées d'une disquette RAM à l'épreuve des effets d'une réinitialisation. Cette disquette, intitulée RAMB0 ou RAD conserve en mémoire les données lorsqu'une réinitialisation du système intervient. Copier Preferences sur RAMB0 peut donc s'avérer extrêmement pratique pour l'utilisateur.

Par ailleurs, vous remarquerez que, trés souvent, certains programmes ne peuvent pas fonctionner en dehors des paramètres fournis par Preferences. C'est le cas, par exemple, de certains traitements de textes qui n'acceptent que 80 colonnes ou de certains jeux qui ne sont exploitables qu'avec un choix de couleurs particulier. Le paragraphe suivant montre comment lire et définir les paramètres de Preferences sans qu'il soit necéssaire de charger le programme.

7.1.1. Lecture et définitions des paramètres de Preferences

Nous allons tout d'abord détailler la structure Preferences sous forme de liste pour illustrer trés précisément ce qui se produit lors de la configuration de ses différents paramètres. La structure Preferences est présentée ici sous forme d'un code source C. A ceux d'entre vous qui ne connaissent pas le langage C, nous conseillons d'apprendre ce langage de programmation dès que possible. Observons , à présent, les 5 colonnes qui constituent le tableau ci-dessous. La première colonne contient les valeurs hexadécimales correspondant à l'emplacement des fonctions C, la seconde colonne donne les équivalences décimales de ces valeurs, puis viennent le type et le nom de la fonction. La dernière colonne, enfin, contient un commentaire destiné à vous aider à comprendre le reste du tableau :

```
Hex     Dec     Type    Nom             Explication

struct  Preferences
{

0x000   000     BYTE    FontHeight;     Hauteur police: Topaz 8/9
0x001   001     UBYTE   PrinterPort;    $00=parallèle, $01=série
0x002   002     USHORT  BaudRate;       Débit en bauds: 00$=110,
                                        $01=300, $02=1200,
                                        $03=2400, $04=4800,
                                        $05=9600, $06=19200,
                                        $07=MIDI
0x004   004     struct  timeval
                        KeyRptSpeed;    Vitesse de répétition
clavier
0x004   004     ULONG   tv_secs;        en secondes
0x008   008     ULONG   tv_micro;       en microsecondes
0x00C   012     struct  timeval
                        KeyRptDelay     Délai répétition clavier
0x00c   012     ULONG   tv_secs;        en secondes
0x010   016     ULONG   tv_micro;       en microsecondes
0x014   020     struct  timeval
                        Doubleclik      Rapidité double clic
0x014   020     ULONG   tv_seconds;     en secondes
0x018   024     ULONG   tv_micro;       en micro_secondes
0x01C   028     USHORT  PointerMatrix36 Tableau graphique
                                        contenant les données du
                                        pointeur de la souris
0x064   100     BYTE    xOffset;        Décalage horiz. du bit de
                                        clic actif
0x065   101     BYTE    yOffset;        Décalage vert.du bit de
                                        clic actif
0x066   102     USHORT  color17;        Couleur 1 du sprite de la
                                        valeur RGB
0x068   104     USHORT  color18;        Couleur 2 du sprite de la
                                        valeur RGB
0x06A   106     USHORT  color19;        Couleur 3 du sprite de la
                                        valeur RGB
0x06c   108     USHORT  PointerTicks;   Sensibilité du pointeur de
                                        la souris à un clic
0x06E   110     USHORT  color0;         Valeur RGB de Workbench
                                        registre de couleur 0
0x070   112     USHORT  color1;         Valeur RGB de Workbench
                                        registre de couleur 1
0x072   114     USHORT  color2;         Valeur RGB de Workbench
                                        registre de couleur 2
0x074   116     USHORT  color3;         Valeur RGB de Workbench
                                        registre de couleur 3
0x076   118     OCTET   ViewXOffest;    Offset X de l'angle sup.
                                        de l'écran Workbench
0x077   119     BYTE    ViewYOffset;    Offset Y de l'angle sup.
                                        de l'écran Workbench
```

```
0x078   120     WORD    ViewInitX;          Valeur d'initialisation de
                                            ViewXOffset
0x07A   122     WORD    ViewInitY;          Valeur d'initialisation de
                                            ViewYOffset
0x07C   124     BOOL    EnableCLI;          Etat du CLI: $00= ON
                                            $01=OFF
0x07D   126     USHORT  PrinterType;        Type d'imprimante
0x080   128     UBYTE   PrinterFilename
                        [FILENAME_SIZE];Nom de fichier du driver
                                            d'imprimante personnalisé
0x09E   158     USHORT  PrintPitch;         Pas de la police:
                                            $000=Pica
                                            $400=Elite, $800=Fine
0x0A0   160     USHORT  PrintQuality;       Qualité de l'impression:
                                            $000=brouillon,
                                            $100=Lettre
0x0A2   162     USHORT  PrintSpacing;       Espacement des lignes:
                                            $000=6 lignes/pouce,
                                            $200= 8 lignes/pouces
0x0A4   164     UWORD   PrintLeftMargin;Marge de gauche
0x0A6   166     UWORD   PrintRightMargin;Marge de droite
0x0A8   168     USHORT  PrintImage;         image imprimée
                                            $00=positive
                                            $01=négative
0x0AA   170     USHORT  PrintAspect;        Sens de l'impression:
                                            $00=horizontal
                                            $01=vertical
0x0AC   172     USHORT  PrintShade;         Nuance: $00=noir et blanc
                                            $01= degradé de gris
                                            $02= couleur
0x0AE   174     WORD    PrintThreshold; Degré de contraste
0x0B0   176     USHORT  Papersize;          Dimension du papier:
                                            $00=Format Lettre US
                                            $10=Format 21,6x35,6
                                            $20=Papier continu étroit
                                            $30=Papier continu large
                                            $40=Attribué par
                                            l'utilisateur
0x0B2   178     UWORD   PaperLength;        Longueur papier en lignes
0x0B4   180     USHORT  PaperType;          Type du papier:$00=Continu
                                            $80=Feuilles séparées

            ***Fin de la version 1.1***

0x0B6   182     UBYTE   SerRWBits;         Nombre de bits
                                            lecture/écriture
0x0B7   183     UBYTE   SerStopBuf;        Nb de bits de stop
0x0B8   184     UBYTE   SerParShk          Parité et mode de
                                            synchronisation
0x0B9   185     UBYTE   LaceWB; Etat de Workbench Interlace
                                            $00=normal,$01=Interlace
0x0BA   186     UBYTE   WorkName[30];       Tampon du nom de
                                            l'imprimante
```

```
0x0BB   187     BYTE    RowSizeChange;  Longueur de ligne définie
                                        par l'utilisateur
0x0BC   188     BYTE    ColumnSizeChange;Nb de colonnes défini par
                                        l'utilisateur

***Fin de la version 1.2

0x0BE   190     UWORD   PrintFlags;
0x0C0   192     UWORD   PrintMaxWidth;  Longueur imprimable
                                        maximum en dixièmes de
                                        pouce.
0x0C2   194     UWORD   PrintMaxHeight; Hauteur imprimable maximum
                                        en dixièmes de pouces.
0x0C4   196     UBYTE   PrintDensity;   Densité de l'impression
0x0C6   198     UWORD   wb_Width;       Largeur de l'écran
                                        Workbench
0x0C8   200     UWORD   wb_Height;      Hauteur de l'écran
                                        Workbench
0x0CA   202     UBYTE   wb_Depth;       Profondeur de l'écran
                                        Workbench
0x0CB   203     BYTE    ext_size;       Extension de la structure
                                        Preferences
0x0CD   204

/**** Fin de la version 1.3****/
```

Vous connaissez, à présent, les données auxquelles la structure Preferences donne accès mais vous ne savez toujours pas comment les utiliser. Il serait interessant de pouvoir lire les paramètres sélectionnés, de les modifier et de sauvegarder les modifications.

C'est ce que le programme ci-dessous se propose d'accomplir. Ce programme permet d'accéder facilement à Preferences sous BASIC à l'aide des fonctions Intuition. Il montre également, au travers de deux exemples simples, comment l'utilisateur peut lire la structure sans difficulté.

Remarque : Les possesseurs du Workbench 1.3 doivent utiliser ce programme avec prudence car cette version est dotée de champs de données étendus. Ce programme fonctionne aussi avec le Kickstart 1.2.

Deux fonctions d'Intuition nous permettent d'atteindre l'objectif poursuivi. L'une des fonctions copie les valeurs courantes dans un emplacement mémoire réservé par le programme, à partir duquel elles pourront être modifiées. La seconde fonction transfère les nouveaux paramètres dans Preferences afin de les enregistrer.

Le programme suivant modifie les couleurs de l'écran Workbench. Les contraintes liées à la mise en page du présent ouvrage ont entrainé la coupure de certaines lignes de BASIC en deux. Les caractères (¶) qui figurent dans le programme ont été ajoutés uniquement pour marquer la fin d'une ligne de programme et, à ce titre, ne doivent pas être reproduits.

```
'*********************************¶
'*                               *¶
'* Add Preferences Data          *¶
'* ----------------------------- *¶
'*                               *¶
'* Auteur : Wolf-Gideon Bleek    *¶
'* Date   : May 15 '88           *¶
'* Nom    : Add-Pref.bas         *¶
'* Version: 1.1                  *¶
'* Système d'expl.: V1.2 & V1.   *¶
'*                               *¶
'*********************************¶
LIBRARY "T&A2:bmaps/exec.library"¶
DECLARE FUNCTION AllocMem& LIBRARY¶
LIBRARY "T&A2:bmaps/intuition.library"¶
Mainprogramm:¶
  ' Preferences chargé¶
  GetPreferences Prefs&, 220&¶
  IF Prefs& = 0 THEN GOTO Ende¶
¶
  ' Entrer les nouvelles couleurs¶
¶
  Colour0 = 110¶
  Colour1 = 112¶
  Colour2 = 114¶
¶
  POKEW Prefs& + Colour0, 1*15+16*4+256*15¶
  POKEW Prefs& + Colour1, 1*15+16*15+256*0¶
  POKEW Prefs& + Colour2, 1*0+16*15+256*8¶
  ' Sauvegarder Preferences¶
  SetPreferences Prefs&¶
Ende:¶
  FreePreferences Prefs&¶
  LIBRARY CLOSE¶
END¶
SUB GetPreferences (Address&, Size&) STATIC¶
  Address& = AllocMem&(Size&+4, 65536&)¶
    IF Address& <> 0 THEN¶
      POKEL Address&, Size&¶
      Address& = Address&+4¶
      CALL GetPrefs(Address&, Size&)¶
    ELSE¶
      Addresse = 0¶
    END IF¶
END SUB¶
```

```
SUB SetPreferences (Address&) STATIC¶
    IF Address& <> 0 THEN¶
      Size& = PEEKL(Address&-4)¶
      CALL SetPrefs(Address&, Size&, -1)¶
    END IF¶
END SUB¶
SUB FreePreferences (Address&) STATIC¶
    IF Address& <> 0 THEN¶
      Size& = PEEKL (Address&-4)¶
      CALL FreeMem(Address&-4, Size&+4)¶
    END IF¶
END SUB¶
```

Description du programme

Le programme se compose de trois sous-programmes qui appellent les fonctions Intuition requises. La fonction GetPreference alloue l'espace mémoire nécessaire à la structure de données et y copie les valeurs appropriées. Si le programme rencontre une erreur, il renvoie la valeur 0. Si tout fonctionne correctement, l'utilisateur peut changer tous les paramètres du programme. Assurez vous de l'exactitude des adresses de départ (cf. tableau ci-dessus) ainsi que la valeur de POKE: POKE correspond à un octet, POKEW à un mot (deux octets) et POKEL à un double-mot (quatre octets). Vous obtiendrez dans le tableau les valeurs qui vous permettront de déterminer la différence entre l'adresse de base et la nouvelle adresse.

La seconde fonction SetPreferences ôte les données de la mémoire du système et informe tous les programmes en cours d'exécution de ce changement au moyen de l'identificateur -1. Si vous préférez que les autres tâches ne soient pas informées des modifications réalisées, vous devez insérer la valeur 0.

La dernière fonction FreePreferences libère la mémoire tampon et la mémoire occupée par Preferences. Ceci est important et témoigne d'un programme bien écrit. Il faut systématiquement libérer la mémoire non employée pour les autres programmes une fois qu'elle ne vous est plus d'aucune utilité.

7.1.2. La nouvelle version de Preferences (Version 1.3)

Toute discussion du système d'exploitation serait incomplète si nous omettions de mentionner les améliorations apportées au programme Preferences. Les améliorations les plus tangibles concernent le contrôle de l'impression. Dans la nouvelle version, les drivers d'imprimantes s'exécutent beaucoup plus rapidement. Une impression réalisée auparavant en dix minutes n'en demande plus que deux.

D'autre part, l'impression graphique à été perfectionnée. Jusque là, Preferences supportait l'impression graphique dans une certaine mesure, mais pas aussi bien qu'on aurait pu l'espérer. La version 1.3 remédie à ces limitations. Avant d'aborder dans les détails l'impression graphique, nous allons passer en revue les autres nouveautés.

L'écran Workbench.

A première vue, les modifications du titre de l'écran ne semblent pas trés impressionantes. Le cadre CLI a été supprimé. Vous pouvez désormais travailler dans le CLI sans avoir à cliquer sur le cadre permettant de l'activer (changement qui s'est longtemps fait attendre).

Une autre amélioration interne ,peu perceptible, concerne l'horloge. Un paramètre est transféré à l'horloge en temps réél ,fonctionnant sur batterie, de façon à ce qu'elle soit réglée chaque fois que vous sélectionnez le cadre Save dans cette fenêtre. Un avantage inappréciable pour tous les possesseurs d'un Amiga 500 ou 2000.

Une "entrée de service" dans Preferences

Une "entrée de service", comparable aux portes ménageant un accès secondaire à l'arrière d'un immeuble, a été ajoutée à Preferences afin d'accélerer les modifications.

Prenons un exemple. Vous avez fait l'acquisition d'une nouvelle imprimante et vous avez besoin de tester les paramètres séléctionnés pour le driver jusqu'à l'obtention d'une combinaison correcte.Pour ce faire, il faut généralement sélectionner Preferences, changer les options dans l'écran Change Printer puis quitter Preferences. A la longue, le caractère

répétitif de toutes ces manipulations devient assommant : sélectionner Preferences et Change Printer, changer les paramètres, quitter Preferences, tester l'impression, sélectionner Preferences, et ainsi de suite.

Workbench 1.3 enregistre Preferences dans un tiroir appelé Prefs. Pour ouvrir ce tiroir, il faut cliquer deux fois dessus. L'icône Preferences et quatre autres icônes s'affichent à l'écran. Trois de ces icônes ménagent un accès direct à certaines parties de Preferences évitant ainsi le passage obligé par l'écran Preferences. Ces icônes sont les suivantes :

`Pointer`	Accès direct à la fenêtre Edit Pointer.
`Printer`	Accès direct à la fenêtre Change Printer.
`Serial`	Accès direct à la fenêtre Change Serial.

Sélectionnez l'une de ces icônes. Sélectionnez ensuite la rubrique Info dans le menu Workbench pour consulter l'écran Info de l'icône. Le cadre Default Tool indique Sys:Prefs/Preferences; Tool Types indique Prefs=Icon_name (Icon_name représente Printer, Serial ou Pointer). Ces spécifications informent le Workbench qu'il faut invoquer la fenêtre demandée à des fins de modification. En court-circuitant Preferences de cette façon, vous gagnerez du temps. Si vous désirez lancer Preferences à partir du CLI, vous pouvez spécifier l'un des arguments indiqués ci-dessous, à la suite du terme Preferences:

```
Preferences [pointer|printer|serial]¶
```

Deux autres icônes sont disponibles dans le tiroir Preferences. Vous aurez souvent recours à l'une de ces icônes alors que l'autre ne vous servira presque jamais. Il s'agit de :

`Preferences`

Accède au programme Preferences

`CopyPrefs`

Copie la configuration mise au point dans Preferences dans DF0:DEVS (utile pour les disques durs qui ne s'initialisent pas automatiquement).

L'impression de graphiques.

Cliquez deux fois sur l'icône Printer du tiroir Prefs. La fenêtre Change Printer apparaît. Cette fenêtre contient deux cadres de sélection des modes graphiques: Graphic 1 et Graphic 2.

Graphic 1

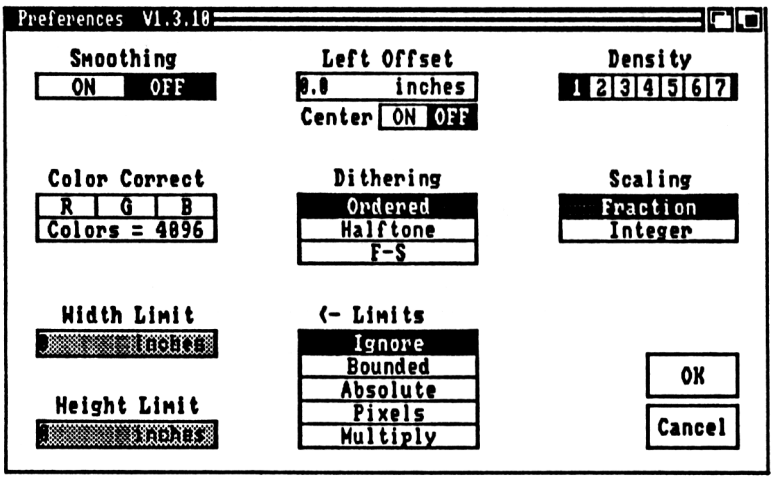
Cliquer sur Graphic 1 revient à cliquer sur Graphic Select dans la fenêtre Change Printer du Workbench 1.2. Une fenêtre présentant différents paramètres d''impression apparaît : Aspect, Image, Shade et Threshold. L'option Shade n'existait pas dans le Workbench 1.2.

Gray Scale2

Donne un dégradé de quatre tons de gris (le moniteur Commodore-Amiga A2024 fournit ce dégradé).

Graphic 2

Un clic sur Graphic 2 nous amène dans une nouvelle fenêtre. Cette fenêtre offre une liste d'options permettant d'améliorer considérablement l'impression graphique.



"Ecran Graphic 2 de Preference 1.3"

Smoothing

Cette fonction donne un aspect plus régulier aux lignes diagonales. Dans certaines applications graphiques d'Amiga les lignes diagonales présentent souvent un aspect dentelé. Cliquez sur ON pour activer la fonction et sur OFF pour la désactiver. La fonction Smoothing est positionnée sur OFF par défaut.

Remarque Smoothing et F-S Dithering (voir plus loin) ne peuvent être activés simultanément. Si vous optez pour F-S puis pour Smoothing ON, le mode Dithering se transforme en Ordered.

Left Offset

Observez à présent l'option immédiatement à droite. Le paramètre Left Offset définit la marge de gauche du graphique (si vous en voulez une). Si vous introduisez 0.0 pouces (valeur implicite), aucun décalage ne se produit.

Remarque le choix de l'option Center ON inactive Left Offset.

Density

Ce paramètre contrôle la densité de l'impression (noirceur). Une densité élevée demande un temps d'impression plus long. Inversement, plus la densité est basse, plus l'impression est rapide. Le paramètre par défaut est 1.

Color Correct

Ce paramètre permet d'atténuer l'intensité des couleurs à l'impression. Les colorations de rouge (R), bleue (B) et vert (G) apparaissent sur l'imprimante comme sur l'écran. Normalement, le nombre de couleurs est de 4096. Toutefois, cette valeur peut être diminuée. Color correct produit généralement une impression de meilleure qualité. Par défaut, aucune option n'est sélectionnée.

Dithering

Cette fonction permet aussi d'intervenir sur les couleurs. Des points de différentes couleurs sont imprimés de façon rapprochées pour n'apparaître à l'oeil nu que comme une seule et même couleur. Il existe trois options :

`Ordered`	Produit une image couleur par points répartis de manière uniforme.
`Halftone`	Produit une image couleur par points en demi-teinte. Ce procédé est utilisé dans l'impression des journaux, des bandes dessinées et autres supports de même type.
`F--S`	F-S mis pour Floyd-Steinberg se réfère à une méthode de distribution des erreurs qui prend un ensemble de points et les répartit de façon aléatoire.Si vous ne possédez pas d'imprimante couleur, il serait intéressant de voir l'effet produit par ce procédé d'impression.

Scaling

Les versions précédentes de Workbench produisaient des graphiques de taille et de format uniques sans tenir aucunement compte de la taille du graphique original. Workbench propose deux options :

`Fraction`	Les graphiques sont réalisés à l'echelle normale comme dans Workbench 1.2.
`Integer`	Cette option permet de définir différentes taille d'impression en fonction des valeurs spécifiées dans les cadres Width Limit et Height Limit.Par exemple, si vous indiquez des uns, le graphique est reproduit à l'echelle 1:1. Des valeurs plus élevées agrandissent l'echelle dans une proportion de deux ou de trois. L'indication d'un 0 dans chaque cadre produit une impression de la taille de la page comme lorsque l'on sélectionne Fraction.

<-Limits

Cette dernière fonction détermine les dimensions maximales du graphique en fonction des valeurs indiquées dans Width Limit et Height Limit. Cinq options sont possibles :

`Ignore`	Ignore les valeurs données et produit un graphique du format spécifié dans l'application employée pour créer le graphique.
`Bounded`	Les dimensions maximales sont celles spécifiées dans les cadres Width Limit et Height Limit. De nombreux graphiques peuvent être imprimés à des dimensions plus réduites par un effet de distorsion.
`Absolute`	Définit les dimensions en valeurs absolues par incréments de 1/10eme de pouce. Par exemple, une Height Limit de 40 et une Width Limit de 50 produisent un graphique de 4 pouces de haut sur 5 pouces de large.
`Pixels`	Pixels définit les limites en pixels au lieu de pouces. Ce mode est similaire à Absolute.
`Multiply`	Ce mode est similaire à Absolute. Toutefois, Multiply permet d'obtenir un graphique dont la dimension originale est multipliée par les valeurs indiquées dans Height Limit et Width Limit. Par exemple, si la valeur de Height Limit est égale à deux et celle de Width Limit égale à 3, le graphique sera deux fois plus haut et trois fois plus large que l'original. Un original de 320x200 donnera un graphique de 640 pixels de largeur sur 600 pixels de largeur.

Nous avons fait le tour des perfectionnements qu'offre Preferences dans la version 1.3. Si vous n'arrivez pas à obtenir tout à fait le résultat escompté, exercez vous encore. Seule la pratique permet d'atteindre à un résulat parfait. Si vous n'avez pas encore acheté d'imprimante, nous vous recommendons de vérifier que les drivers d'imprimantes incorporés à Workbench 1.3 correspondent à l'imprimante que vous convoitez.

Remarque : Si vous ne trouvez pas le driver adapté à votre imprimante, jettez un coup d'oeil dans la disquette Extras. Les développeurs d'Amiga y ont mis quelques drivers qui ,faute de place, ne pouvaient figurer sur la disquette Workbench. Peut être y trouverez-vous votre bonheur.

7.1.3. Le tiroir Utilities dans 1.3

Le champ d'action de la disquette Workbench s'étend au delà de la gestion de l'environnement de travail et des données du système. Elle contient aussi des programmes utilitaires qui fournissent une aide appréciable à l'utilisateur engagé dans un travail de programmation ou autre.

Cliquez deux fois sur le tiroir Utilities pour l'ouvrir. De nombreux programmes disponibles dans la version 1.2 ont été repris dans la version 1.3 après avoir subi quelques transformations. La version 1.3 renferme des paramètres et des options supplémentaires ainsi que des méthodes d'accès au CLI peu connues.

Definition des paramètres de l'horloge à partir du CLI

On peut régler très facilement les paramètres de l'horloge à partir des menus. Cependant, chaque fois que vous réinitialiser Workbench, les paramètres reviennent à leur état initial. Ceux-ci peuvent être sélectionnés à partir de la commande CLI qui s'appelle Clock.

Les arguments de Clock ont le format suivant :

```
CLOCK [ANALOG|DIGITAL1|DIGITAL2][=<x>,<y> [,<width>,
<height>]] [12HOUR|24HOUR] [SECONDS] [DATE]¶
```

Signification des arguments :

```
ANALOG|DIGITAL1|DIGITAL2
```

Indique l'un des trois types d'horloge disponibles sur Workbench (par défaut, ANALOG).

```
<x>,<y>
```

Indique la position de l'horloge sur l'écran. <x> représente le nombre de pixels horizontaux à partir de la bordure gauche de l'écran. <y> représente le nombre de pixels verticaux à partir du haut de l'écran (par défaut 10,15).

```
<width>, <height>
```

Indique la hauteur et la largeur de l'horloge analogique uniquement. Les horloges numériques ignorent ces arguments.

```
12HOUR|24HOUR
```

Indique soit une horloge de 12 heures (AM/PM), soit une horloge de 24 heures (par défaut, horloge de 12 heures).

```
SECONDS
```

Active ou inactive l'affichage des secondes (par défaut=OFF).

```
DATE
```

Active ou inative l'affichage de la date (par défaut= OFF).

La commande CLI suivante affiche l'heure et la date au cours d'une session Workbench :

```
Utilities/Clock DIGITAL2 24HOUR DATE
```

ClockPtr

Vous pouvez exécuter ClockPtr en lançant le programme à partir du CLI ou en cliquant deux fois sur son icône. Lorsque vous accédez à ClockPtr, le pointeur se transforme en horloge numérique toutes les fois qu'il est positionné sur l'écran Workbench. L'horloge ClockPtr peut afficher différents types de paramètres. Si l'on déplace le pointeur sur la gauche de l'écran Workbench, les minutes et les secondes s'affichent. Si, par contre, on déplace le pointeur vers la droite de l'écran, ce sont les heures et les minutes qui s'affichent.

Pour inactiver ClockPtr, appuyez sur <Ctrl><c> si vous l'avez démarré à l'aide de la fonction Execute du CLI. Entrez Break si vous l'avez démarré avec la fonction Run ou cliquez deux fois sur l'icône si vous l'avez lancé à partir de Workbench.

CMD

Ce nouveau programme envoie dans un fichier la sortie de CMD_WRITE qui serait autrement dirigée vers un périphérique de sortie série ou parallèle. Cette commande accroît considérablement le contrôle exercé par l'utilisateur. Un CMD_WRITE transfère des données en direction de l'imprimante que ce soit par l'intermédiaire du CLI ou d'un programme.

Les arguments de CMD ont le format suivant :

```
CMD <devicename><filename>[OPT s<124>m<124>n]
```

Signification des arguments :

```
<devicename><filename>
```

Indique le périphérique série ou parallèle initial (<devicename>) et le nom du fichier vers lequel les informations doivent être transférées (<filename>). PAR: ou SER: ne peuvent être utilisés comme nom de périphérique.

```
OPTs
```

N'exécute pas d'amorce d'écriture (se produit généralement après une réinitialisation). Dans certain drivers d'imprimante, l'amorce d'écriture met l'imprimante sous tension. Ce mode n'est pas utile pour transférer des données à un fichier. Si votre imprimante émet ce code, utilisez s.

```
OPTm
```

Argument utilisé avec plusieurs fichiers jusqu'à ce qu'un Break se produise ou un <Ctrl><C>. Normalement Cmd s'inactive de lui même après l'écriture d'un fichier. Cette option maintient la commande active pour écrire plusieurs fichiers ou réaliser des copies d'un même fichier.

```
OPTn
```

Active le mode "notification". Ce mode tient l'utilisateur informé de la progression du transfert de fichier.

Install Printer

Nous avons évoqué précédemment les drivers d'imprimantes placés sur la disquette Extras par manque d'espace sur la disquette Workbench. Vous vous remémorez aussi sans doute le fichier script InstallPrinter ,configuré par IconX. InstallPrinter copie un driver d'imprimante de la disquette Extras dans le répertoire approprié (le répertoire DEVS:).

Vous pouvez exécuter ce programme à partir du CLI ou en cliquant deux fois sur son icône. Ce programme affiche la liste des drivers d'imprimante disponibles sur la disquette Extras. Entrez le nom du driver voulu en face du message d'invite (pour copier plusieurs drivers, vous pouvez vous servir des caractères de substitution) et appuyez sur <Return>. Ce fichier script copie le driver dans le répertoire DEVS: et ajoute le nom du driver à la liste contenue dans la fenêtre Change Printer de Preferences. Si vous désirez quitter InstallPrinter sans copier aucun driver, enfoncez la touche <Return>.

More

Cette commande affiche les fichiers texte ASCII sur l'écran. More est accessible à partir du CLI ou de Workbench. Dans le CLI, la syntaxe de la commande est la suivante :

```
More <filename>
```

Pour accéder à More à partir de Workbench, il faut cliquer deux fois sur l'icône. Si vous désirez ouvrir un fichier texte symbolisé par une icône, cliquez une fois sur l'icône More, puis maintenez la touche <Shift> enfoncée tout en cliquant deux fois sur l'icône du texte que vous voulez lire.

Pour vous déplacer à l'intérieur d'un fichier texte, actionnez les touches décrites ci-dessous :

<espace>	Page suivante
<Espace arrière>	Page précédente
<Retour>	Ligne suivante
<<>	(inférieur à)première page du fichier
<>>	(supérieur à)dernière page du fichier
<%n>	Affiche le segment n% du fichier
<Ctrl><L>	Efface la fenêtre
</>	Recherche les caractères placés après </> dans un texte en tenant compte des majuscules et des minuscules (par exemple, /March ne trouvera pas march).
<.><string>	Recherche les caractères placés après <.> dans un texte sans tenir compte des majuscules et des minuscules (par exemple, .March localisera March et march).
<n>	Recherche l'occurence suivante d'une chaîne indiquée après un </> ou un <.>
<h>	Liste d'aide
<q>	(ou <Ctrl><C>) pour quitter
<E>	Edition à l'aide de l'éditeur trouvé dans ENV:EDITOR.

More affiche le pourcentage d'un fichier en bas de l'écran. Lorsque la fin du fichier est atteinte , le message devient -Less-

Lorsque vous accédez à More au moyen du CLI ou du Shell, vous pouvez invoquer immédiatement un éditeur pour modifier le fichier que vous consultez à condition que la variable ENV:EDITOR contienne une spécification. Cette variable doit comporter le chemin correct vers l'éditeur (par exemple: C:ED). Si ENV:EDITOR est définie, appuyez sur <Shift><E> pour appeler l'éditeur.

Printfiles

Cette commande imprime des fichiers. Vous avez la possibilité d'imprimer plusieurs fichiers en même temps si vous accédez au fichier à partir de CLI et en fournissant plusieurs spécifications de fichiers.

Printfiles est accessible à partir de Workbench ou du CLI.

Le format de la commande est le suivant :

```
PrintFiles
[-f]<filename>[[-f]<filename>[[-f]<filename>...
```

Signification des arguments

```
-f
```

Active le mode avance papier.Ce mode intercale une avance papier entre chaque fichier. Pour activer ce mode, il faut sélectionner l'écran info de Printfiles dans Workbench et entrer FLAGS=formfeed dans la cadre TOOL TYPES.

```
<filename>
```

Indique le nom du ou des fichiers à imprimer.

Si vous voulez accéder à Printfiles à partir de Workbench, cliquez sur l'icône symbolisant le fichier que vous voulez imprimer. Maintenez la touche <Shift> enfoncée ,cliquez sur l'icône correspondant au fichier à et cliquez deux fois sur l'icône Printfiles.

7.1.3.1. Les autres utilitaires de Workbench 1.3

Graphic Dump

Cette commande effectue une recopie de l'écran d'Intuition placé au premier plan sur l'imprimante. La version 1.3 de Graphic Dump vous permet d'en spécifier la taille.

Vous pouvez accéder à Graphic Dump en cliquant deux fois sur son icône ou à partir de CLI. Les arguments de la commande CLI ont le format suivant :

```
GraphicDump [TINY|SMALL|MEDIUM|LARGE|xdots:ydots]
```

Signification des arguments :

TINY — La recopie d'écran est réduite à un quart de la largeur imprimable (la hauteur s'ajuste proportionnellement).

SMALL — La copie d'écran est réduite à la moitié de la largeur imprimable (la hauteur s'ajuste proportionnellement).

MEDIUM — La recopie d'écran est réduite aux trois-quarts de la largeur imprimable (la hauteur s'ajuste proportionnellement).

LARGE — La copie d'écran est égale à la largeur imprimable (la hauteur s'ajuste proportionnellement). Cette valeur est la valeur par défaut lorsque l'on accède à Graphic Dump à partir de Workbench.

7.2. Les outils de la disquette Extras

Nous avons mentionné, au début de ce chapitre, que la disquette Extras recèle d'autres trésors en dehors d'AmigaBASIC. Vous trouverez sur cette disquette des drivers d'imprimante et un certain nombre de programmes utilitaires supplémentaires.

FED

FED est l'abréviation de Font EDitor. Cette utilitaire vous permet de changer les jeux de caractères existants ou de créer les votres.

Remarque Deux précisions à propos de FED. Tout d'abord, ne transformez jamais les jeux de caractères sur votre disquette Workbench originale. Ensuite, il faut savoir que FED ne peut pas éditer de jeux de caractères dont la dimension est supérieure à 32X32 pixels.

Vous pouvez accéder à FED en cliquant deux fois sur son icône. Une fenêtre contenant des cadres de sélection et des caractères s'affiche. Nous dressons ci-dessous la liste des menus, des rubriques qui les composent et des cadres disponibles:

Project

Ce menu propose des options de gestion de fichier. Ses rubriques sont les suivantes :

New Efface les jeux de caractères disponibles en mémoire.

Open Permet d'ouvrir un jeu de caractères à partir du disque.

Save Enregistre le jeu de caractères courant sur le disque.

Save as Enregistre le jeu de caractères courant sur le disque sous un nouveau nom.

About Ecran d'information qui indique qui a écrit FED.

Quit Quitte FED et vous renvoie au Workbench.

Edit

Ce menu vous permet d'éditer le jeu de caractères dans son intégralité (vous ne pouvez pas modifier une lettre uniquement. De plus, certaines transformations sont irréversibles).

Make italic Met le jeu de caractères en italiques.

Make bold Met le jeu de caractères en gras.

Make Underlined
Met le jeu de caractères en souligné.

Copy to Copie un caractère d'une boîte dans une autre. Sélectionnez le caractère que vous voulez copier puis Copy to. Sélectionnez ensuite la boîte de destination du caractère. Le mode Copy est activé jusqu'à une nouvelle sélection de Copy to.

Erase Efface les caractères dans la boîte sélectionnée.

All right Déplace les caractères vers la droite d'un pixel.

All left Déplace les caractères vers la gauche d'un pixel.

All up Déplace les caractères vers le haut d'un pixel.

All down Déplace les caractères vers le bas d'un pixel.

Attributes

Ce menu permet de sélectionner le format et le style de la police de caractères.

Font type Les options proposées sont Proportional Spacing (espacement proportionnel) et Fixed Width spacing (espacement fixe).

Font style Normal, Italic, Bold (gras), Underlined (souligné), Extended (Etendu).

Rendering Forward ou Reverse

Keytoy2000

Cette utilitaire affiche la configuration générale du clavier. Le premier écran montre le résultat des touches sans l'emploi de la touche <Shift>. Si vous voulez visualiser les touches en combinaison avec les touches <Shift>, <Ctrl>, etc..., cliquez sur la case correspondante de la fenêtre KeyToy ou sur la touche appropriée du clavier.

Voici quelques "règles de base" d'utilisation de KeyToy2000:

➢ Les touches en bleu ne sont pas accessibles par Keytoy (Les touches <CTRL>, <SHIFT> et <ALT> sont accessibles)

➢ Les touches en jaune ne sont pas accessibles (<Caps Lock>, <Commodore> et la touche <Amiga> de droite).

➢ Les touches $$ contiennent des chaînes formées de plusieurs caractères.

➢ Les caractères qui débutent par ^ ou ~ sont des caractères de contrôle.

➢ Les touches en blanc ne sont pas définies.

Memory Display

Trois options sont possibles. La première d'entre elles affiche la mémoire disponible lorsque l'on se trouve dans l'écran Workbench. Ceci présente quelques inconvénients. Premièrement, cet écran vient se superposer sur les autres fenêtres et écrans. De plus, la mémoire disponible n'est visualisable que lorsque l'écran Workbench est actif. La disquette Extras comprend deux utilitaires écrits pour afficher la mémoire disponible.

FreeMap

FreeMap crée un nouvel écran et affiche l'affectation de la mémoire sous forme graphique. Cet écran affiche la CHIP-RAM et la FAST-RAM, cette dernière option étant nouvelle. FreeMap montre egalement la segmentation de la mémoire lorsque plusieurs tâches sont exécutées. La sortie est graphique.

Le menu Info donne une description de la façon dont FreeMap affiche la mémoire. Pour quitter FreeMap, il faut sélectionner la rubrique Please du menu Quit.

PerfMon

PerfMon est un programme de contrôle des performances qui indique la chip-RAM et la fast-RAM disponibles ainsi que le temps processeur utilisé. Il est intéressant de voir l'usage qui est fait du 68000. Une ligne noire indique la configuration idéale. Au nombre des inconvénients de PerfMon, il faut signaler qu'il utilise sa propre fenêtre et chip-RAM ainsi que sa propre fenêtre de texte.

Palette

Cet outil permet de modifier les couleurs de l'écran. Contrairement à Preferences qui se limite aux couleurs de l'écran Workbench, Palette donne accès à toute la gamme de couleurs disponibles sur Amiga. Vous pouvez ouvrir Palette en cliquant deux fois sur son icône ou à partir du CLI.

Les arguments de Palette ont le format suivant :

```
Palette [<bitplane>] [<screentype>]
```

Signification des arguments :

```
bitplanes
```

Spécifie la profondeur du bitplane obtenu par la sélection des nouvelles couleurs de palette. 1 donne deux couleurs, 2 donne 4 couleurs, 3 donne 8 couleurs, 4 donne 16 couleurs et 5, 32 couleurs.

```
Screentype
```

Spécifie la résolution de l'écran en cours de test. 0 donne une résolution de 320X200 pixels; 1 donne une résolution entrelacée de 320X400 pixels; 2 donne une résolution de 640X200 pixels et 3 une résolution entrelacée de 640X400 pixels.

La fenêtre Palette présente plusieurs cadres et sliders qui permettent à l'utilisateur d'intervenir sur les couleurs.

Remarque : Les modifications apportées dans Palette sont temporaires. Vous ne pouvez enregistrer ces informations sur le disque.

MEMACS

MEMACS est un éditeur de texte situé dans le répertoire Tools : de la disquette Extras. Il existe plusieurs versions de MEMACS (abbréviation de MicroEmacs) qui varient en fonction de la marque de l'ordinateur. MEMACS est un éditeur <page> bien plus puissant que ED. Il permet l'édition de plusieurs fichiers simultanément à condition que la RAM disponible soit suffisante pour que plusieurs programmes puissent coexister en mémoire.

Vous pouvez exécuter MEMACS en cliquant deux fois sur l'icône Workbench ou à partir du CLI. Les arguments de MEMACS ont le format suivant :

```
Memacs [<filename>] [goto <n>] [OPT W]
```

Signification des arguments :

```
[<filename>]
```

Spécifie le nom du fichier que vous voulez charger dans MEMACS.

```
[<goto <n>]
```

Déplace le curseur à la ligne n de filename.

```
[OPT W]
```

Place MEMACS dans une fenêtre Workbench.

On peut travailler dans MEMACS avec la souris et le clavier. Les titres des menus sont évidents et les raccourcis clavier listés sur les menus sont les mêmes d'une machine à l'autre. Ainsi, après avoir utilisé la version AMiga de MEMACS, vous pourrez passer sans problème sur la version MEMACS d'IBM ou d'UNIX.

Chapitre 8

Personnalisation du Workbench

Peut-être vous êtes vous demandé comment modifier les messages du Workbench pour leur donner une touche personnelle ou les traduire en langue française. Une telle opération n'est pas aussi aisée que la modification d'un programme en BASIC, mais elle demeure possible.

Ce chapitre montre comment les possesseurs d'un Amiga 1000 peuvent intervenir sur les textes de Workbench et les changer. Tout ce dont vous avez besoin est une copie de sauvegarde de la disquette Kickstart et d'un gestionnaire de disque. Utilisez des copies de Kickstart et de Workbench ! N'utilisez SURTOUT PAS vos originaux lors de l'édition de fichiers importants.

Si vous n'avez jamais utilisé un gestionnaire de disque,nous vous rappelons que ces programmes vous permettent de lire, de modifier et d'enregistrer des blocs de mémoire sur un disque. Si vous ne possédez pas de gestionnaire de disque, procurez vous en un dans le domaine public. De nombreux programmes de ce genre sont disponibles sur le marché comme, par exemple, Diskmon ou TSEditor. Si vous préférez, faites l'acquisition du gestionnaire de disque fourni dans Le Livre du Lecteur de Disquettes. Apprenez les principes fondamentaux de l'utilisation d'un gestionnaire de disque avant de vous lancer dans ce projet. Une simple erreur de frappe peut provoquer la destruction des données.

Remarque Les instructions contenues dans le présent chapitre concernent uniquement la disquette KickStart de l'Amiga 1000. Ceux d'entre vous qui possèdent un Amiga 500 ou

un Amiga 2000 ont un Kickstart incorporé dans la mémoire morte (ROM), ce qui complique quelque peu les choses. Si vous désirez malgré tout modifier votre ROM Kickstart, vous devez en extraire le code à l'aide d'un moniteur de langage machine, le changer et le graver dans une nouvelle EPROM (mémoire morte reprogrammable). L'aide d'une personne maitrisant la programmation de mémoire morte et le langage machine vous sera probablement utile si vous n'êtes pas, vous-mêmes, un expert en la matière. Les utilisateurs d'un Amiga 500 ou 2000 préféreront sans doute sauter ce chapitre. Il pourront cependant en retirer de nombreuses informations sur le fonctionnement du Kickstart.

8.1. Les préparatifs

Passons une dernière fois en revue les outils nécessaires à l'opération que l'on se propose d'effectuer. Vous aurez besoin des éléments suivants :

Copies de sauvegarde de Kickstart

Nous ne cesserons de le répéter, n'utilisez pas la disquette Kickstart originale pour réaliser le présent projet. Prenez votre disquette Kickstart, protégez la en écriture et faites en une ou deux sauvegardes. Utilisez ,pour ce faire, la rubrique Duplicate du menu Workbench ou la commande Diskcopy du CLI. Une fois les sauvegardes accomplies, mettez l'original de côté. Travaillez à partir de l'une des deux copies de sauvegarde et gardez la seconde copie à portée de la main en cas de destruction accidentelle de la disquette que vous utilisez. En fait, il serait plus sage d'effectuer plus de deux copies.

Un gestionnaire de disque

Nous l'avons déjà dit, les gestionnaires de disque lisent les données à partir d'un disque (généralement par tranches de secteurs). Ces données peuvent être lues, modifiées puis réécrites sur le disque.

Vous pouvez vous procurer des programmes de gestion de disque en différents endroits. Beaucoup de disques tombés dans le domaine public sont équipés de gestionnaires de disque. De même vous pourrez obtenir une liste de gestionnaires en C, en AmigaBASIC ou en langage machine dans de nombreux ouvrages.

L'une des caractéristiques essentielles d'un gestionnaire de disque est de permettre l'affichage des données sous la forme de caractères ASCII et l'écriture d' un texte sur un bloc de disque. Le programme de gestion de disque donné dans l'ouvrage d'Abacus intitulé Amiga Disk Drives Inside and Out a l'avantage de présenter ces caractéristiques. A ce titre, il nous a servi de base de travail lors de la préparation du présent ouvrage et de la modification des disquette Kickstart.

La disquette Workbench ne vous sera d'aucune utilité dans le cadre de ce projet. Vous aurez peut-être la curiosité de rechercher les textes concernés dans la disquette Workbench. Si vous ne l'avez pas déjà fait, nous vous prévenons dès à présent que c'est inutile; ces textes ne s'y trouvent pas. La disquette Kickstart (ou ROM) contient tous les messages appartenant au système d'exploitation. Ces messages sont chargés au cours du processus d'initialisation. Vous ne ferez appel à la disquette Workbench que pour tester votre nouveau système.

Voici quelques règles de base à ne pas négliger lorsque l'on entreprend de modifier une disquette KickStart :

❶ Travaillez sur des copies toutes les fois que vous écrivez des blocs de disque à l'aide d'un gestionnaire de disque. Si vous employez les originaux, vous encourez le risque de détruire des données importantes. Certaines zones de disque ne doivent pas être trafiquées; une seul erreur peut rendre un disque inutilisable et bon uniquement à être reformaté.

❷ Lorsque vous écrivez sur un disque, faites une réinitialisation de test, assez souvent au début (l'idéal serait d'en effectuer une aprés chaque modification). De cette façon, si un problème surgit vous pourrez au moins en déterminer la cause. Lorsque vous aurez plus confiance en vous, vous pourrez espacer le rythme des réinitialisations.

❸ Utilisez des versions compatibles de Workbench et de Kickstart (Kickstart 1.2 avec Workbench 1.2 et Kickstart 1.3 avec Workbench 1.3). Intervertir les versions peut être source de problèmes.

❹ Lors d'une réinitialisation avec un Kickstart modifié, l'Amiga effectue un total de contrôle interne pour vérifier si le Kickstart est compatible avec lui même. Sur une disquette Kickstart "normale" l'icône symbolisant une main du Workbench est invoquée à chaque réinitialisation de l'ordinateur. La version modifiée génère un message système demandant l'insertion de la disquette Kickstart. Ceci est parfaitement normal. Gardez les deux disquettes Workbench et Kickstart à portée de main au cours d'une initialisation ou d'un réinitialisation.

❺ Lorsque vous utilisez un gestionnaire de disque, surtout prenez votre temps. Un espace ou un caractère placés à mauvais escient peuvent se solder par une défaillance du système. Observez avec attention la structure d'un texte et ne changez rien à sa longueur. Si vous brûlez les étapes, les risques de commettre une erreur grave augmentent. Lenteur et prudence sont ici de mise.

Voilà pour les recommendations. Passons à présent aux choses sérieuses.

8.2. Les premières étapes de la modification d'un texte

Les principes de base étant posés, il est temps de se lancer dans le processus de transformation. La liste des messages que nous voulons modifier est donnée dans le paragraphe 8.3.

8.2.1. Les messages de démarrage et la fenêtre AmigaDOS

Initialisez votre gestionnaire de disque et faites les actions nécessaires pour lire votre copie de sauvegarde de Kickstart. Lisez le bloc 454 décimal ($01c5 en hexadécimal) sur la version 1.2 de Kickstart et le bloc 453 décimal ($01c4 en hexadécimal) sur la version 1.3. Recherchez les noms système LIBS, DEVS ou FONTS (selon la version de Kickstart que vous possédez, un défilement vertical sera necéssaire pour localiser les blocs mentionnés). Recherchez à présent le texte ci-dessous dans le bloc en question ou dans le bloc suivant:

```
CON:0/0/640/200/AmigaDOS
```

Cette ligne ouvre une fenêtre console intitulée AmigaDOS. C'est la première chose que l'on aperçoit lorsque Workbench est à l'oeuvre. Cette ligne crée une fenêtre au cours de l'initialisation, avant que l'écran AmigaDOS ne s'affiche. La configuration de cette commande définit la hauteur de la fenêtre à 200 pixels, ce qui correspond aux normes américaines (vidéo NTSC = National Television system Committee). Comme vous possédez un écran PAL (conforme aux normes européennes), vous allez transformer la valeur Y de 200 en 256. Ou si vous préférez, vous avez la possibilité de réduire la taille de votre fenêtre AmigaDOS. Remplacons, par exemple, 200 par 150. La ligne ainsi modifiée se présente comme suit:

```
CON:0/0/640/150/AmigaDOS
```

Effectuez les modifications et réécrivez le bloc sur le disque. Quittez le gestionnaire de disque. Eteignez l'ordinateur pendant 10 secondes environ. Rallumez le et essayez de l'intialiser avec la diquette Kickstart. L'Amiga vous demandera d'introduire la disquette Workbench. Insérez la disquette comme d'habitude. Une fenêtre AmigaDOS réduite aux 3/4 doit s'afficher sur l'écran. Voilà qui n'est pas si extraordinaire. Mais attendez la suite.

Eteignez l'ordinateur pendant 10 secondes et ôtez les disquettes d'essai. Rallumez l'appareil et réinitialisez en vous servant des disquettes que vous utilisez normalement. Exécutez à nouveau le programme de gestion de fichier.

Réalisons, à présent, notre première vrai modification ; changer le nom de la fenêtre AmigaDOS, par exemple. Lisez le bloc que vous venez de modifier (454 décimal ou $01c5 héxadécimal dans la majeure partie des versions 1.2 eet 453 décimal ou $01c4 dans les versions 1.3). Examinez à nouveau la ligne suivante (vous venez de modifier cette ligne de façon à réduire les dimensions de la fenêtre AmigaDOS):

```
CON:0/0/640/150/AmigaDOS
```

Le département de conception de produits chérit tout particulièrement un utilisateur qui porte le nom de Elmer. Elmer est un personnage célèbre du monde des médias depuis les années 30. Ce personnage est caractérisé par

un accent que l'on peut qualifier de ...bizarre. Nous avons changé le titre de la fenêtre AmigaDOS de façon à personaliser le DOS à l'intention de ce fameux Elmer, ce qui sera beaucoup plus drôle que la simple traduction des messages en français à laquelle vous deviez sans doute vous attendre. Vous pouvez baptiser la fenêtre du nom de votre choix. La seule contrainte concerne la longueur du nom : il doit être de la même longueur ou plus court que le nom original. S'il est plus court, ajustez le à la longueur correcte à l'aide d'espaces. Voici quelques exemples de substitution correctes et incorrectes:

Original

```
CON:0/0/640/150/AmigaDOS
```

Correct

```
CON:0/0/640/150/FredDOS
```

Correct

```
CON:0/0/640/150/Doug_DOS
```

Incorrect

```
CON:0/0/640/150/Richard_DOS
```

AmigaDOS occupe 8 caractères. Modifier le titre de la fenêtre de la façon suivante :

```
CON:0/0/640/150/ElmerDOS
```

Ecrivez le bloc sur le disque. Quittez le gestionnaire de disque. Eteignez l'ordinateur pendant 10 secondes. Allumez le à nouveau et essayez de l'initialiser à partir de la disquette Kickstart modifiée. L'Amiga vous demandera d'introduire la disquette Workbench. Insérez la disquette comme d'habitude. Une fenêtre DOS intitulée ElmerDOS doit apparaître à l'écran. Si le système a une défaillance, éteignez l'ordinateur et réinitialisez à partir des disquettes normales Kickstart et Workbench.

Eteignez l'ordinateur pendant 10 secondes et otez les disquettes d'essai. Rallumez le et réinitialisez à l'aide des disquettes Workbench et Kickstart que vous utilisez normalement. Exécutez à nouveau le programme de gestion du disque.

Lisez les même blocs que précédemment (454 décimal etc...). Changeons maintenant les messages de copyright. Recherchez l'un des textes suivants:

```
Copyright . 1985, 1986 Commodore-Amiga¶
Copyright . 1988 Commodore-Amiga¶
```

Modifiez le texte comme indiqué ci-dessous (ne touchez pas au point qui s'intercale entre le terme Copyright et l'année- c'est un code machine et non pas un simple point):

```
Copywight . 1988, 1989 Commodore-Amiga¶
Copywight . 1989 Commodore-Amiga¶
```

Ecrivez ce bloc sur le disque comme auparavant. A présent, tournez vous vers le texte ci-dessous (il se trouve peut-être dans le bloc suivant):

```
All rights reserved..Release 1.2¶
All rights reserved..Release 1.3¶
```

Modifiez le comme suit :

```
All wights weserved..Wewease 1.2¶
All wights weserved..Wewease 1.3¶
```

Ecrivez ce bloc sur le disque. Eteignez l'ordinateur pendant 10 secondes et faites une initialisation d'essai. Examinez attentivement l'écran et appréciez le résultat de vos efforts. Félicitations; vous êtes en face d'une version expérimentale ElmerDOS !

Comme nous l'avons précisé plus haut, vous pourrez changer les messages à votre gré plus tard (la seule limite à respecter étant leur longueur), mais pour l'instant suivez nos indications à la lettre. Cela peut sembler stupide de prime abord, mais une fois le procédé assimilé, vous pourrez le mettre en oeuvre facilement à votre tour.

8.2.2. Modification de la barre de titre et des menus.

La plupart des fonctions de Workbench sont accessibles à partir de menus. Vous pouvez aussi changer les titres des menus et les rubriques qui les composent, ou les traduire à l'intention des personnes qui parlent une langue étrangère.

Reprenons le concept ElmerDos là où nous l'avions laissé. Initialisez le gestionnaire de disque; les textes des menus débutent au bloc 384 décimal dans Kickstart 1.2 et au bloc 383 décimal dans la version 1.3. Vous trouverez à cet endroit, le titre du menu Workbench et ses rubriques. Les autre titres et rubriques de menus se trouvent dans ce même bloc et dans le bloc suivant. Avant de procéder à leur substitution, nous vous rappelons que la longueur des nouveaux noms ne doit pas excéder celle du nom d'origine, et que les noms plus courts doivent être ajustés par l'insertion d'espaces.

Avant de poursuivre, nous allons dire deux mots de la dernière rubrique du menu Special. Si vous regardez quelques caractères plus loin, vous noterez la présence d'un certain nombre de blancs. Ceci est en fait un autre menu qui intervient lorsqu'une erreur se produit. L'Amiga se branche alors sur un "debugger" dans la ROM Kickstart qui permet à l'utilisateur d'examiner l'erreur en détail, à condition toutefois que le terminal soit connecté à l'interface RS-232 de l'ordinateur.

Vous pouvez activer ce menu et appeler les données à partir de là. Ceci n'a d'interêt que pour un programmeur et ne concerne nullement un utilisateur moyen.

Pour utiliser ce menu caché, il faut ajouter le terme Debug à la commande loadWB dans la séquence de démarrage. Le mot Debug doit aussi être inséré dans les espaces blancs. Pour l'instant, laissons ce sujet de côté pour nous concentrer sur les menus qui intéressent un utilisateur amateur. Pour obtenir un complément d'informations sur le menu Debug et sur ROM-Wack, reportez-vous au livre de l'AmigaDOS pour la version 1.3.

8.2.3. Du nouveau dans les rubriques de menus et dans les messages

Vous ambitionnez peut être de traduire les titres et les rubriques de menus en français ou bien tout simplement de leur donner un libellé plus amusant.

En parcourant les blocs du disque, vous vous apercevrez que l'Amiga contient d'innombrables rubriques de menus et messages. Nous concentrerons notre attention sur les plus importants d'entre eux. Pour le reste, vous trouverez toute une liste à la fin de ce paragraphe que vous aurez le loisir d'explorer.

Toutes les fois que vous travaillez avec votre Amiga, l'écran Workbench est actif. Seulement, vous ne pouvez utiliser Workbench que si la séquence de démarrage a eté chargée avec LoadWB. Dans tous les cas, le barre de titre indique Workbench Screen.

Poursuivons notre processus de personalisation de Kickstart pour ElmerDOS. Sélectionnez le bloc 247 dans Kickstart 1.2 (bloc 245 dans la version 1.3). Recherchez la ligne:

```
Workbench Screen
```

Modifiez la comme suit :

```
Wuhkbench Scween
```

Ecrivez le bloc sur le disque selon la procédure habituelle. Réinitialisez pour voir si le nouveau Kickstart tient la route. Si tout va bien, nous pouvons continuer.

La barre de titre de l'écran Workbench réside dans le bloc 359. Elle se présente de la façon suivante:

```
Workbench release 1.2.  %ld free memory
```

Soyez extrêment prudent avec ce texte! Le point, les deux espaces, %&d et encore espace sont des instructions spécifiques à C. Ces instructions allouent un espace au système pour insérer le nombre d'octets libres. Ces caractères doivent rester intacts! Maintenant, changez cette ligne pour l'harmoniser à notre Kickstart ElmerDOS:

```
Wuhbench wewease 1.2. %ld fwee memowy
```

Ecrivez le bloc sur le disque.

Au tour des rubriques de menu, maintenant. Nous vous rappelons que les textes des menus débutent au bloc 384 décimal (Kickstart 1.2) ou au bloc 383 décimal (version 1.3). Le menu de Workbench a l'aspect suivant:

```
Workbench.Open.Close.Duplicate.Rename.Info.Discard..
```

Transformez le en ElmerDOS comme indiqué ci-dessous (deux espaces ont été introduits pour équilibrer le remplacement de Discard):

```
Wuhkbench.Opun.Cwose.Dupwicate.Wename..Huh?..Bwast
..
```

Ecrivez le bloc sur le disque.

Le titre du menu Disk, maintenant. Il se présente comme suit:

```
Disk..Empty Trash.Initialize
```

Changez le de façon à lire:

```
Disk..Empty Twash.Initiawize
```

Encore une fois, ne touchez surtout pas aux points qui séparent le titre de menu et les différentes rubriques.

Le menu Special d'origine (qui peut s'étendre sur deux blocs) est le suivant :

```
Special.Clean Up..Last
Error..Redraw..Snapshot..Version
```

Transformez le de façon à obtenir :

```
pecial.Cwean Up..Wast
Ewwow..Wedraw..Snapshot..Wuhersion
```

Ecrivez le bloc sur le disque et effectuez une réinitialisation pour tester l'ensemble. Déroulez les menus pour voir le résultat obtenu.

Pendant que nous y sommes, modifions aussi la zone de dialogue invoquée par Discard (Bwast). Allez au bloc 385 (Kickstart 1.2) ou au bloc 383 (Kickstart 1.3). Voici le texte original:

```
Warning: you..cannot get back.what you discard!.
```

Voici le nouveau texte (laissez les points tranquilles):

```
Warning: you..cannot get back.what you bwast! .
```

Ecrivez le bloc sur le disque.

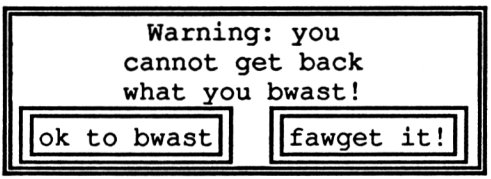
La dernière transformation concerne les cadres de la zone de dialogue. Allez au bloc 381 (Kickstart 1.2) ou au bloc 379 (Kickstart 1.3). Vous y localiserez les textes qui composent la zone de dialogue Discard:

```
ok to discard.forget it!
```

Remplacez ce texte par cet autre (notez les espaces de remplissage aprés bwast):

```
ok to bwast  .fawget it!
```

Ecrivez le bloc sur le disque. Réinitialisez pour vérifier le résultat produit. Ouvez l'icône de la disquette Workbench. Cliquez une fois sur une icône de fichier et sélectionnez la rubrique Bwast dans le menu Wuhkbench. La zone de dialogue suivante s'affiche:



Cliquez sur fawget it! pour quitter sans supprimer le fichier.

Vous avez à présent une vue d'ensemble des procédures de modification de la disquette Kickstart. La phase d'initiation sur l'exemple ElmerDOS est terminée. A vous maintenant d'adapter maitenant Kickstart à votre goût.

8.3. Guide des messages de Kickstart

La liste ci-dessous énumère les messages présents sur la disquette Kickstart. La première colonne précise les numéros de bloc correspondant à 1.2 (pour les autres versions, il est necéssaire de descendre ou de remonter d'un ou deux blocs). La seconde colonne donne le texte proprement dit. Un dernier rappel : le texte de substitution de doit pas dépasser la longueur du texte original. Les textes plus courts doivent être remplis par des espaces ou des caractères de ponctuation.

Certains de ces messages sont invisibles si on n'y accède pas par l'intermédaire d'un gestionnaire de disque. Nous vous en faisons part tout de même pour que vous sachiez ce qui se dissimule sur votre disquette.

Bloc	Message
25- 26	Not enough memory .&.Software failure. .&.Recoverable Alert Press left mouse button to continue Guru Meditation # ...
26	ROM WACK commands: alter.boot.clear.fill.find.go. ig.limit.list.regs.reset.resume.set.show.user
56	The Amiga Wizards brings this power to you
245	=LEFT BUTTON OK=.....=RIGHT BUTTON CANCEL= System Request
247	Workbench Screen..WBenchPort
263-	Brought to you by not a mere Wizard
292	DMC68343 FLOATING POINT FIRMWARE(C) COPYRIGHT 1981 BY MOTOROLA INC.
341	read error %ld
348	Please insert volume..in any drive Retry.Cancel
349	Error while ...
351	examining directory.accessing %s.opening %s. reading.writing.moving %s.removing %s.examining %s..writing %s.
353	This drawer is not really a directory
358	Can't rename this disk
359	Workbench release 1.2. %ld free memory
373	SYS:System/Diskcopy

Bloc	Message
375	Icons cannot be moved into this window
376	This disk cannot be copied
378	'%s' cannot be moved out of its window.. Disks are of incompatible type and cannot be copied
381	OK to discard.forget it!
384	Onfo not available Workbench Open Close Duplicate Rename Info Discard
384 385	Disk Empty Trash Initialize Special Clean Up Last Error Redraw SnapShot Version Debug.flushlibs Warning: you..cannot get back .what you discard
391	Cannot read info file %s
394	Enter the new name. Press return when done... Do not use '%lc' in names Ran Out of memory. Please free some and try again
396	DOS
401	The icon(s) have no default tool
407	This drawer cannot be opened
418	kodiak andy carl Initial CLI
422- 423	Disk corrupt - task held Software error - task held Finish ALL disk activity Select CANCEL to reset/debug
425	Retry.Cancel
426	Fault.Error Code %N
428	Volume..in Unit Ox!!.. is not validated is write protected.. Please replace volume... in any drive.... is full..... Not a DOS disk.. No disk present..... has a read/write error You MUST replace volume
432	Please insert volume.... in any drive

Bloc	Message
454- 455	SYS.:...LIBS....DEVS....FONTS...C...L...S...DFX.CLI Restart.FileHandler. CON:0/0640/200/AmigaDOS Copyright . 1985
456	CON.RAW.SER.PAR.PRT devs:system-configuration
460	*** BREAK-CLI c:..Unknown command %s Unable to load %s .%S failed return code %N Error in command name
462- 463	Too many > or < command too long Unable to open redirection file. Syntax error CLI error: %s
488	BUSY
510	Out of memory Cannot open disc device :L/Disk-Validator...L:Disk-Validator
511- 512	Out of range already set Error validation disk Disk structure corrupt Use DiskDoctor to correct it out of range Disk is unreadable checksum error
880	Empty
882	Trashcan
883	Trashcan.Info

Vous n'êtes pas sans avoir remarqué les indicateurs %ld,%s,etc... Ce sont des spécificateurs de format C qui doivent rester invariables. Il constituent des points de référence pour insérer du texte et des données numériques.

Annexe A

Messages d'erreur

Cette annexe donne la liste des messages d'erreur dans l'ordre de leur numérotation.

103 insufficient free store

AmigaDOS ne peut charger le programme car la mémoire disponible est insuffisante. Il faut mettre fin à une autre tâche ou fermer des fenêtres CLI.

105 task table full

AmigaDOS ne peut gérer que 20 tâches CLI simultanément. Lorsque ce nombre est dépassé, aucune autre fenêtre CLI ne peut être ouverte.

120 argument line invalid or too long

Ce message apparaît lorsque les paramètres ajoutés à une commande sont incorrects.

121 file is not an object module

Seuls les fichiers programmes peuvent être exécutés directement à partir de l'indication de leur seul nom (exemple: les fichiers script doivent débuter par Execute).

122 invalid resident library during load

Un problème s'est produit lors de l'ouverture ou du chargement d'une bibliothèque.

202 object in use

Ce message empêche l'écriture sur un fichier ou la suppression d'un répertoire quand ceux ci sont utilisés par une autre tâche.

203 object already exists

Un nom donné existe déjà et ne peut pas être effacé

204 directory not found

AmigaDOS ne peut pas trouver le répertoire indiqué

205 object not found

AmigaDOS ne peut pas trouver un fichier ou un répertoire

206 invalid window description

Syntaxe incorrecte lors de l'ouverture d'une fenêtre. Vérifier les coordonnées de la fenêtre et la syntaxe (exemple:con:0/0/635/100/.)

209 packet request type unknown

Un driver n'est pas en mesure de réaliser un accès demandé. Ce message est dû uniquement à des erreurs de programmation.

211 invalid object lock

Une erreur de programmation a entrainé un code de verrouillage non valide.

212 object not of the required type

Cette erreur est provoquée par une confusion entre des fichiers et des répertoires.

213 disk not validated

La disquette insérée dans le lecteur est probablement endommagée.

214 disk write-protected

L'accès en écriture sur ce disque est impossible. Le taquet de protection en écriture est probablement mal positionné.

215 rename across devices attempted

Rename ne peut pas renommer d'une disquette à une autre.

216 directory not empty

Tentative de suppression d' un répertoire non vide.

218 device (or volume) not mounted

AmigaDOS ne peut pas trouver la disquette indiquée.

219 seek failure

L'argument ajouté à la fonction seek est incorrect.

220 comment too big

Les commentaires affectés à un fichier par FileNote ne peuvent excéder 80 caractères.

221 disk full

Il n'y a pas assez d'espace sur le disque pour réaliser l'action désirée.

222 file is protected from deletion

Le fichier est probablement protégé contre l'effacement avec Protect.

223 file is write protected

224 file is read protected

Ces deux commandes réagissent aux drapeaux de protection définis à l'aide de Protect. Ces deux messages d'erreur ne sont plus utilisés car la version actuelle du DOS supporte uniquement le drapeau D.

225 not a valid DOS disk

La structure de disque est complètement détruite ou n'a pas été formatée sous AmigaDOS.

226 no disk in drive

Aucun disque ne se trouve dans le lecteur utilisé

232 no more entries in directory

Cette erreur de programmation signale que la routine ExNext ne trouve pas d'autres entrées dans un répertoire.

Annexe B

Les raccourcis CLI

Les commandes CLI et Shell comprennent des séquences de commandes <Ctrl> et <Esc> qui peuvent être entrées à partir du clavier. Les caractères de commandes peuvent aussi être utilisés dans des fichiers scripts par l'intermédiaire de la commande Echo. La commande d'échappement (Escape) est alors inscrite entre guillemets et débute par une astérisque qui représente la touche <Esc> (par exemple : echo "*ec" efface l'écran). De cette façon, on peut modifier le type de caractère, choisir une couleur, déplacer le curseur, etc..

<Esc>c	Efface l'écran et inactive les modes spéciaux
<Esc>[0m	Inactive tous les modes spéciaux
<Esc>[1m	Caractères gras
<Esc>[2m	Ecriture noire (n de couleur 2)
<Esc>[3m	Italiques
<Esc>[30m	Ecriture bleue (n de couleur 0)
<Esc>[31m	Ecriture blanche (n de couleur 1)
<Esc>[32m	Ecriture noire (n de couleur 2)
<Esc>[33m	Ecriture orange (n de couleur 3)
<Esc>[4m	Souligné
<Esc>[40m	Fond bleu (n de couleur 0)
<Esc>[41m	Fond blanc(numéro de couleur 1)
<Esc>[42m	Fond noir (numéro de couleur 2)
<Esc>[43m	Fond orange(numéro de couleur 3)
<Esc>[7m	Vidéo inverse (normalement bleu sur blanc)
<Esc>[8m	Ecriture bleue, invisible(ou numéro de couleur 0)
<Esc>[nu	Largeur en caractères de la fenêtre CLI (n)
<Esc>[nx	Marge gauche de la fenêtre CLI en pixels (n)

<Esc>[ny	Distance en pixels du bord supérieur de la fenêtre (n)
<Esc>[nt	Nombre de lignes de la fenêtre CLI(n)
<Crtl><H>	Efface le dernier caractère introduit (ou <Backspace>
<Crtl><I>	Déplace le caractère d'une tabulation vers la droite ou <Tab> (par défaut, cinq caractères).
<Crtl><J>	Effectue un saut de ligne sans exécuter la commande entrée permettant ainsi la saisie de plusieurs commandes à la suite. Si l'on appuie sur <Return>, toutes les commandes sont exécutées les unes à la suite des autres.
<Crtl><K>	Déplace le curseur d'une position vers le haut. Le texte ne peut être modifié.
<Crtl><L>	Efface l'écran
<Crtl><M>	Fin de ligne. Les commandes entrées sont exécutées. Equivalent à <Return>.
<Crtl><N>	Active le jeu de caractères Alt. Seuls les caractères spéciaux sont imprimés.
<Crtl><O>	Inactive le jeu de caractères Alt et revient au jeu de caractères normal.
<Crtl><X>	Efface l'intégralité de la ligne courante;
<Crtl><\>	Marque la fin d'un fichier dans AmigaDOS. Met également fin aux entrées dans les fenêtres Con:.

Annexe C

Séquences Escape d'impression

Les séquences Escape suivantes sont transcrites à partir des drivers d'imprimante intégrés dans Preferences.

Séquence Escape	Signification
<Esc>c	Initialise l'imprimante
<Esc>#1	Inactive les autres modes
<Esc>D	Saut de ligne
<Esc>E	Saut de ligne + Retour chariot
<Esc>M	Retour d'une ligne
<Esc>[Om	Caractères normaux
<Esc>[1m	Caractères gras
<Esc>[22m	Suppression des caractères gras
<Esc>[3m	Italiques
<Esc>[23m	Suppression des caractères italiques
<Esc>[4m	Souligné
<Esc>[24m	Suppression de souligné
<Esc>[xm	Couleurs (x = 30 à 39 [premier plan] ou 40 à 49 [Fond])
<Esc>[Ow	Impression normale
<Esc>[2w	Police Elite
<Esc>[1w	Suppression de la police Elite
<Esc>[4w	Mode condensé
<Esc>[3w	Suppression du mode condensé

<Esc>[6w	Mode étendu
<Esc>[5w	Suppression du mode étendu
<Esc>[2"z	NLQ
<Esc>[1"z	Suppression de NLQ
<Esc>[4"z	Double frappe
<Esc>[3"z	Suppression de la double frappe
<Esc>[6"z	Ombré
<Esc>[5"z	Suppression de ombré
<Esc>[2v	Exposant
<Esc>[1v	Suppression de exposant
<Esc>[4v	Indice
<Esc>[3v	Suppression de indice
<Esc>[0v	Retour aux caractères normaux
<Esc>[2p	Espacement proportionnel
<Esc>[1p	Suppression de l'espacement proportionnel
<Esc>[Op	Effacement de l'espacement proportionnel
<Esc>[xE	Espacement proportionnel = x
<Esc>[5F	Alignement à gauche
<Esc>[7F	Alignement à droite
<Esc>[6F	Justification
<Esc>[0F	Suppression de la justification
<Esc>[3F	Ajustement de la largeur des lettres
<Esc>[1F	Justification centrée
<Esc>[0z	Interligne de 1/8 de pouce
<Esc>[1z	Interligne de 1/6 de pouce
<Esc>[xt	Longueur de la page = x
<Esc>[xq	Saut des perforations défini à x lignes
<Esc>[0q	Suppression du saut de perforation
<Esc>[(B	Jeu de caractères américain
<Esc>[(R	Jeu de caractères français
<Esc>[(K	Jeu de caractères allemand
<Esc>[(A	Jeu de caractères anglais
<Esc>[(E	Jeu de caractères danois (n1)
<Esc>[(H	Jeu de caractères suédois
<Esc>[(Y	Jeu de caractères italien
<Esc>[(Z	Jeu de caractères espagnol
<Esc>[(J	Jeu de caractères japonais
<Esc>[(6	Jeu de caractères norvégien

<Esc>[(C	Jeu de caractères danois (n2)
<Esc>[#9	Position de la marge de gauche
<Esc>[#0	Position de la marge de droite
<Esc>[#8	Position de l'en-tête
<Esc>[#2	Position du pied de page
<Esc>[#3	Suppression des marges
<Esc>[xyr	En-tête à x lignes du haut et pied de page à y lignes du bas.
<Esc>[xys	Position de la marge de gauche (x) et de la marge de droite (y).
<Esc>[H	Positionnement d'une tabulation horizontale.
<Esc>[J	Positionnement d'une tabulation verticale.
<Esc>[0g	Suppression d'un taquet de tabulation horizontale
<Esc>[3g	Suppression de toutes les tabulations horizontales
<Esc>[1g	Suppression des tabulations verticales
<Esc>[4g	Suppression de toutes les tabulations verticales.
<Esc>[#4	Suppression de toutes les tabulations
<Esc>[#5	Positionnement des taquets de tabulation standard.

Annexe D

Les codes de Guru Meditation

Les Guru Meditations vous informent des défaillances du système. Deux nombres de huit chiffres sont alors renvoyés. Le premier de ces nombres fournit des informations détaillées sur l'erreur. Il se décompose comme suit:

Code d'identification système	Catégorie d'erreur	Nº de l'erreur
xx	xx	xxxx

Codes d'identification système

00	CPU Trap
01	Bibliothèque Exec
02	Bibliothèque Graphics
03	Bibliothèque Layers
04	Bibliothèque Intuition
05	Bibliothèque Math
06	Bibliothèque CList
07	Bibliothèque DOS
08	Bibliothèque RAM
09	Bibliothèque Icon
0A	Bibliothèque Expansion
10	Périphérique Audio
11	Périphérique Gameport
13	Périphérique Keyboard
14	Périphérique Trackdisk

15	Périphérique Timer
20	Ressource CIA
21	Ressource Disk
22	Rssource Misc
30	Bootstrap
31	Workbench
32	Diskcopy

Catégorie d'erreur

01	Mémoire insuffisante
02	Erreur due à MakeLibrary
03	Erreur due à OpenLibrary
04	Erreur due à OpenDevice
05	Erreur due à OpenResource
06	Erreur d'entrée/sortie
07	Pas de signal

Codes spéciaux de Guru Meditation

Lorsqu'un code d'identification système débute par un nombre supérieur ou égal à 8, l'erreur est irréparable. Il faut soustraire 8 au premier chiffre pour obtenir le véritable code d'identification.

CPU Traps

00000002	Erreur de bus
00000003	Erreur d'adresse
00000004	Instruction illégale
00000005	Division par zéro
00000006	Instruction CHK
00000007	Instruction TRAPV
00000008	Violation de privilège
00000009	Trace
0000000A	Code opération 1010
0000000B	Code opération 1111

Bibliothèque Exec

81000001	Erreur dans le total de contrôle du vecteur d'exception du 68000
81000002	Erreur dans le total de contrôle de ExecBase
81000003	Erreur dans le total de contrôle d'un bibliothèque
81000004	Mémoire insuffisante pour MakeLibrary
81000005	Liste mémoire indéchiffrable
81000006	Pas de mémoire libre pour le serveur d'interruption
81000007	Problème avec InitAPtr
81000008	Sémaphore indéchiffrable
81000009	"Bogus Exception"

Bibliothèque Graphics

8210001	Mémoire insuffisante pour Copper display list
8210002	Mémoire insuffisante pour Copper command list
8210003	Dépassement de Copper list
8210004	Dépassement de Copper Intermediate
8210005	Mémoire insuffisante pour l'en-tête de Copper list
8210006	Manque de mémoire pour Long Frame
8210007	Manque de mémoire pour Short Frame
8210008	Mémoire insuffisante pour Flood Fill
8210009	Mémoire insuffisante pour TmpRas
821000A	Mémoire insuffisante pour BltBitMap
821000B	"Region Memory"

Bibliothèque Layers

83010001	Pas de mémoire disponible pour les layers

Bibliothèque Intuition

84000001	Type du gadget inconnu
84100002	Mémoire insuffisante pour créer un port
84100003	Mémoire insuffisante pour Item Plane Alloc
84100004	Mémoire insuffisante pour Sub Alloc
84100005	Mémoire insuffisante pour Plane Alloc
84100006	Coordonnées initiales inférieures à RelZero
84100007	Mémoire insuffisante pour ouvrir un écran

84100008	Mémoire insuffisante pour Raster Alloc
84100009	Type inconnu lors de Open Sys Screen
8410000A	Mémoire insuffisantes pour les gadgets
8410000B	Mémoire insuffisante pour la fenêtre
8410000C	Code de retour erroné rencontré dans Intuition
8410000D	IDCMP a transmis un message erroné
8410000E	Réponse incompréhensible
8410000F	Erreur lors de l'ouverture de Console

Bibliothèque DOS

07010001	Problème de mémoire au démarrage
07010002	Problème avec EndTask
07010003	Problème avec Qpkt
07010004	Paquet reçu n'était pas attendu
07010005	Problème avec Free Vec
07010006	Erreur dans l'ordre des DiskBlocks
07010007	Bitmap erroné
07010008	Key déjà effacée
07010009	Total de contrôle incorrect
0701000A	Erreur de disquette
0701000B	Valeur incorrect pour Key
0701000C	Problème lors de overlay

Bibliothèque RAM

08000001	Segment-List erroné

Bibliothèque Expansion

0A000001	Problème lors de Expansion Free

Trackdisk

14000001	Erreur en recherche à l'étalonnage
14000002	Erreur de retard du timer

Timer

15000001	Demande incorrecte
15000002	Transfert incorrect

Ressource disque

21000001	Unité de lecture a préparé la disquette
21000002	Interruption : unité inactive

Initialisation

30000001	Erreur dans le code d'initialisation

Index

!

<-Limits 7-195
68010 ..... 4-143, 4-147
68020 4-150
68030 ..... 4-150 - 4-151
68851 4-151
68881 4-151
68882 4-152
6888x 4-150

A

AllocMem 3-126
AllocMem() 2-31
AmigaBASIC 1-15
AmigaDOS 2-90
Ascenceurs 2-57

B

BCPL 3-129
BootPri 5-155
BufMemType 5-155

C

Clear 1-18
Clock 7-196
ClockPtr 7-197
Close() 2-90
CloseWindow 2-35
CMD 7-198
Codes de Guru Meditation D-233
Color Correct 7-193
ColorCycle 3-131
Commande Assign 5-153
Commande Install 3-121
Commande Mount 5-153
Commande SetAlert 4-146
CON: 1-10, 5-154
ConvertFD 2-29
Copy ..... 1-12 - 1-13
CurrentDir() 2-103

D

DeciGel 4-146
DeleteFile() 2-106
Density 7-193
Device 5-155
DF1:NODOS 5-158
Dir 1-10, 1-13
Disquette Extras 7-183
Disquette Workbench 7-183
Dithering 7-194
DIVS 3-116
DIVU 3-116
DoIO& 6-179
Dos.bmap 2-89
Dos.library ..... 2-89 - 2-90
Dos_lib.fd 2-89
DrawBorder() 2-59

E

Echo 1-10
Ed 1-10
ElmerDOS ..... 8-214 - 8-215
Examine() 2-98
Exec.bmap 2-29, 2-98
Exec.library 2-29, 2-31
Exec_lib.fd 2-29
ExNext() 2-100

F

FastFileSystem 5-161
FED 7-202
Fichier .info 2-106
FindTask() 1-19
Flags 5-155
Floyd-Steinberg 7-194
Fonction Lock 2-97
Fonctions mathématiques 5-163
Fraction 7-194
FreeMap 7-205

G

Gadgets 2-32
Gadgets proportionnels 2-64
GetDiskObject 1-16
GetMsg() 2-35
Graphic 1 7-192
Graphic 2 7-192
Graphic Dump 7-201
Graphics_lib.fd 2-29

H

Halftone 7-194
HighCyl 5-158

I

Impressions graphiques 6-174
Info() 2-103
Install Printer 7-199
Integer 7-194
Intuition 2-32
Intuition.library 2-29
Intuition_lib.fd 2-29

J

JAM2 2-40

K

Keytoy2000 7-204

L

Langage C 3-115
Left Offset 7-193
List 1-10, 1-13
LoadSeg 3-128

LoadSegment 3-136
LoadWB 1-14
LowCyl 5-158
LPRINT 2-84

M

MEMACS 7-206
Messages d'erreur A-223
Mode Superviseur 3-116
More 7-199
MOVES 4-149

N

NewCon 1-13, 5-161
Nombres en virgule flottante double précision 5-166

O

Open() 2-90
OpenDevice() 6-173
OpenWindow() 2-32, 2-35

P

Palette 7-205
PAR: 1-20
ParentDir() 2-103
PerfMon 7-205
Périphérique PIPE 5-159

Pointeur BCPL 3-129
POKE 2-40
PolyDraw() 2-59
Preferences 1-24, 7-183
Printer.device 6-167
Printfiles 7-201
PRT: 1-20

R

Raccourcis CLI B-227
RAM-Disk ..... 1-12 - 1-13
RAM:BasicClip 1-18
RAMB0 1-13
RAW: 5-154
RectangleFill 2-52
RectFill 2-52
Rename 1-13
Rename() 2-106
Répertoire DEVS: 5-154
Répertoire T: 1-9
Reserved 5-155
RTD 4-149
RTE 3-116

S

Scaling 7-194
Séquences Escape d'impression C-229
SER: 1-20
SetAlert 3-116
SetDrawMode 2-51
SetDrMd 2-51
SetProtection() 2-107
SetTaskPri() 1-19
Smoothing 7-193
Startup-Sequence 1-9

Structure IORequest 6-179
Structure NewWindow 2-32
Structure Preferences 7-184
Structure Window 2-32
Surfaces 5-155
System-Configuration 1-23

T

Tiroir Utilities 7-196
Topaz 5-158
Type 1-10

U

Unit 5-155
UnloadSeg 3-128

X

XClose() 2-92
XOpen() 2-92
XRead() 2-92
XWrite() 2-92

Dans la même collection...

Référence	Titre		Prix T.T.C.
...			
...			
ML197	Bien débuter avec l'AMIGA		149.00
ML553	Bien débuter en C sur AMIGA		149.00
GL123	Guide SOS AMIGADOS/AMIGABASIC		149.00
ML634	Le livre du lecteur de disquette	disquette incluse	299.00
ML512	La Bible de l'AMIGA		299.00
ML612	La Bible de l'AMIGA	disquette incluse	399.00
ML632	Le livre de la musique	disquette incluse	199.00
ML505	Le Livre du Graphisme		249.00
ML605	Le Livre du Graphisme	disquette incluse	349.00
ML504	Le Grand livre de l'AMIGABASIC		249.00
ML604	Le Grand livre de l'AMIGABASIC	disquette incluse	349.00
ML513	Le Livre de l'AMIGADOS		199.00
ML562	Le livre du GFA Basic Amiga		149.00
ML662	Le livre du GFA Basic Amiga	disquette incluse	249.00
ML198	Le Livre du Langage Machine		199.00
ML188	Trucs et Astuces AMIGA		199.00
ML288	Trucs et Astuces AMIGA	disquette incluse	299.00
...			
...			

Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94)
Dépôt légal - Août 1989
N° d'impression: 5140

AMIGA

TRUCS ET ASTUCES

NOUVELLE EDITION ENTIEREMENT REACTUALISEE

BLEEK / MAELGER / WELTNER

Une foule d'astuces inédites sur les toutes nouvelles commandes de l'Amiga, et de nombreux trucs qui apporteront à vos applications une indéniable note d'originalité. TRUCS ET ASTUCES vous explique comment gagner du temps en utilisant AmigaDOS et WorkBench 1.3 de façon plus efficace, et comment affiner et améliorer vos techniques de programmation avec le CLI et AmigaBASIC. Vous pourrez aussi créer un environnement de programmation convivial au moyen de menus déroulants, de sliders et de tableaux. De plus cet ouvrage vous dévoile comment écrire votre programme de détection de virus, désactiver la fast-RAM, équiper votre machine d'un processeur plus rapide, utiliser NewCon et PIPE, accéder sous Basic à des programmes en langage machine et C, et bien plus encore.

Principaux sujets traités :

- Un programme de détection de virus.
- Les nouveaux périphériques de l'AmigaDOS et Preferences 1.3.
- Des trucs d'utilisation pour les nouveaux utilitaires de la disquette Extras.
- Comment personnaliser KickStart pour les possesseurs d'un Amiga 1000.
- Comment améliorer le Basic au moyen de ColorCyle et du programme désactivation de la souris.
- Comment désactiver la fast-RAM et les lecteurs de disquettes.

9 782868 992208

REF. ML 570. 129 F.
REF. ML 670. 229 F avec la disquette.
ISBN 2-86899-220-X / ISSN 0980-1928

EDITIONS MICRO APPLICATION
58 RUE DU FAUBOURG POISSONNIERE 75010 PARIS
TEL (1) 47 70 32 44

TRUCS ET ASTUCES

AMIGA